E.02.35-ComplexityCosts-D2.2-Model Implementation

Document information	
Project Title	ComplexityCosts
Project Number	E.02.35
Project Manager	University of Westminster
Deliverable Name	Model Implementation
Deliverable ID	D2.2
Edition	01.01.00
Template Version	03.00.00

Task contributors
University of Westminster
Fundación Instituto de Investigación Innaxis (Innaxis)

Abstract

The primary objective of ComplexityCosts is to better understand ATM network performance trade-offs for different stakeholder investment mechanisms under certain disruptions. In this report, software implementation of the ComplexityCosts model is reported. Classes, methods and interface functions are presented with practical examples. Previous deliverables on the ComplexityCosts sub-models on passengers and traffic, mechanisms and disturbances, are updated.

Authoring & Approval

Prepared By - *Authors of the document.*		
Name & Company	Position & Title	Date
Samuel Cristóbal / Innaxis	Consortium Member	25AUG15
Luis Delgado / University of Westminster	Consortium Member	25AUG15
Graham Tanner / University of Westminster	Consortium Member	25AUG15
Andrew Cook / University of Westminster	Project Leader	25AUG15
David Pérez / Innaxis	Consortium Member	25AUG15
Adeline de Montlaur / Universitat Politècnica de Catalunya	Visiting Researcher (UoW)	25AUG15

Reviewed By - *Reviewers internal to the project.*		
Name & Company	Position & Title	Date
David Pérez / Innaxis	Consortium Member	26AUG15
Andrew Cook / University of Westminster	Project Leader	26AUG15

Reviewed By - *Other SESAR projects, Airspace Users, staff association, military, Industrial Support, other organisations.*		
Name & Company	Position & Title	Date
N/A		

Approved for submission to the SJU By - *Representatives of the company involved in the project.*		
Name & Company	Position & Title	Date
Samuel Cristóbal / Innaxis	Consortium Member	27AUG15

Rejected By - *Representatives of the company involved in the project.*		
Name & Company	Position & Title	Date
N/A		

Rational for rejection
None.

Document History

Edition	Date	Status	Author	Justification
01.00.00	28AUG15	Release	Samuel Cristóbal	New document for review by EUROCONTROL
01.01.00	26OCT15	Review	Samuel Cristóbal	Incorporating EUROCONTROL comments

Intellectual Property Rights (foreground)

This deliverable consists of Foreground owned by one or several Members or their Affiliates.

Table of Contents

List of tables

List of figures

Executive summary

The primary objective of ComplexityCosts is to better understand ATM network performance trade-offs for different stakeholder investment mechanisms in the context of uncertainty, under different types of disturbance. A variety of investment mechanisms for affording network resilience and robustness will be considered. ComplexityCosts seeks to quantify, and improve the understanding of, complex interdependencies that are often overlooked in trade-off models.

The ComplexityCosts simulation platform makes use of state of the art in simulation techniques, a blend between soft-computing, event-driven programming, cloud-based infrastructure and elements from agent-based modelling make the ComplexityCosts model one of a kind. Soft-computing techniques enable the model to work with uncertainty and incomplete information whilst the event-driven paradigm grants the software to have higher level of flexibility, never seen before in a more classical sequential programming. Program flow is determined by events, algorithms of shorten length involving only a limited number of actors in a restricted language.

New 2014 passenger itineraries are being developed for the project's passenger and traffic model, building on an existing in-house 2010 dataset. Each itinerary will assign a passenger to individual flights, with up to two connections, along with ticket price and premium/non-premium seat class distinction. Maximum seating capacity per aircraft cannot be exceeded whilst total passengers and overall average load factor constraints are respected. Assignment algorithms can already reassign a large proportion of existing 2010 itineraries onto 2014 flights.

The modelling of the mechanisms (increasing ATCO hours in selected sectors, dynamic cost indexing and A-CDM with improved airline passenger reaccommodation policies) includes how the mechanisms affect flight operations, the stakeholders uptake and the strategic and tactical costs of implementation. The benefit of increasing ATCO in selected sectors is directly related with the modelling of the disruptions of staff shortage; the cost will be estimated at a tactical (cost of ATCO hour) and at a strategic level (ATCO training). Dynamic cost indexing will be based on modelling the cost of individual flights at different speeds at different phases of the flight. A-CDM will imply more predictable airport operations and improvements on the minimum turnaround time required at the airport. Changes to airline reaccommodation policies will be modelled as passenger wait/no-wait rules and enhanced algorithms to reaccommodate passengers with missed connections. Literature research and suppliers' cost data will be used.

Three types of disturbance will be modelled: staff shortage disruptions, industrial actions and weather at airports. In order to model these disruptions their scope and intensity should be considered. Staff shortage and industrial action models are based on the analysis of the ATFM regulations. Weather-related disturbances will be based on a particular disruptive weather day in Europe generating both ATFM delay and tactical departure and arrival delays.

A dedicated section on verification and credibility assessment is presented. Verification and validation are defined and clearly differentiated. Various levels of modelling are described, and appropriate validation is identified in terms of the project's Technology Readiness Level (TRL). Early stages of validation, performed at TRL2, should be understood as a credibility assessment, is explained. Detailed actions supporting the model's credibility assessment and verification are presented. The corresponding importance of the integrity of the model input data, across multiple components of the project, including its sourcing, cleaning and reviewing, are discussed.

In order to be able to assess the mechanisms under the disturbances, it is necessary to model the corresponding cost impacts. These are modelled as costs of delay to the airline. The main tactical costs of delay are comprised of passenger, fuel, maintenance and crew costs. We present a summary of the results to date of the tactical delay cost modelling, which has substantially progressed since the previous deliverable. Although the passenger cost of delay is often a dominating delay cost for operators, there remains limited evidence supporting the calculation of such costs. These costs have therefore gone out to airline consultation, running from 18 August 2015 – 02 October 2015.

Future plans for the delay costs to be published as a separate, stand-alone reference publication are also presented. The report concludes with a review of the next steps for the project, broken down by the wider model implementation, the mechanism and disturbance modelling, and the delay cost estimations. Key forthcoming deliverables are also identified.

1 Introduction

1.1 Purpose of the document

The primary objective of ComplexityCosts is to better understand ATM network performance trade-offs for different stakeholder investment in the context of uncertainty. A variety of investment mechanisms which afford network resilience and robustness will be considered. The project's stochastic, layered network model will include interacting elements and feedback loops by embracing the non-linearities of complexity and significantly advancing the state of the art.

The following are the key objectives of this deliverable:

- illustrate key aspects of the model implementation;
- provide the fundamentals to understand the ComplexityCosts model from a software developer perspective;
- explain how the ComplexityCosts software is deployed using preconfigured scripts;
- update progress on the ComplexityCosts sub-models on passengers and traffic, mechanisms and disturbances;
- address the verification and credibility of the tool;
- update the cost of delay modelling.

It is worth noting that providing a commented source code for the ComplexityCosts software tool is not an objective of this deliverable.

1.2 Intended readership

This report is primarily intended for internal usage. A background in computational sciences and software development would be useful for the technical texts.

1.3 Inputs from other projects

Not applicable.

1.4 Glossary of terms

Not applicable.

1.5 Acronyms and Terminology

Term	Definition
ABM	Agent-based modelling
ACC	Area Control Centre
A-CDM	Airport Collaborative Decision Making
AIRAC	Aeronautical Information Regulation And Control
ANSP	Air Navigation Service Provider
AO	Airline Operator
APU	Auxiliary Power Unit

Term	Definition
ATC	Air Traffic Control
ATCO/ATCo	Air Traffic Controller
ATFM	Air Traffic Flow Management
ATM	Air Traffic Management
AU	Airspace User(s)
AWS	Amazon web services
BADA	Base of Aircraft Data
CC	ComplexityCosts
CSV	Comma-Separated Values
DCI	Dynamic cost index
D*X.Y*	Deliverable X.Y (Number Y within Workpackage X)
ECTL / EUROCONTROL	European Organisation for the Safety of Air Navigation
EIBT	Estimated in-block time
E-OCVM	European Operational Concept Validation Methodology
ETA	Estimated time of arrival
ETOT	Estimated take-off time
GCD	Great circle distance
GDS	Global Distribution System (a system that distributes inventory on behalf of airlines)
IAF	Initial Approach Fix
ICAO	International Civil Aviation Organization
INX	Innaxis Foundation and Research Institute
MCT	Minimum Connecting Time
MTOW	Maximum take-off weight
O/D	Origin and Destination
OOP	Object-oriented programming
PTI	Passing Time to IAF

Term	Definition
SESAR	Single European Sky ATM Research
SJU	SESAR Joint Undertaking
SOBT/SIBT	Scheduled off/in block-time
TRB	Transportation Research Board
TRLs	Technology Readiness Level(s)
TV	Traffic Volume
UoW	University of Westminster
V&V	Validation and verification
vCPU	(Virtual) Central Processor Unit
WP	Workpackage
XML	eXtensible Mark-up Language

2 Model Overview

2.1 ComplexityCosts framework elements

This section contains an overall description of the ComplexityCosts tool. A more conceptual description of ComplexityCosts can be found in previous deliverables. The ComplexityCosts tool is a stochastic layered network model, implemented using soft-computing and event-driven techniques and deployed in a cloud-based infrastructure. There are four key elements:

- Data Store, a database, file or any other source of information (e.g. web-service, SOAP) containing the scenario information.
- Event-stack manager, handles the events and drives the simulation flow.
- Environment, a collection of actors, their functionalities and status.
- Events, sub-processes triggered by other actors that make use of environment information and functions

The relations and work-flow between the elements are described in Figure 1. First, scenarios are loaded from the data store and the event stack and environment are both initialized. Then, in an iterative process, events stored in the stack are processed in order until it is empty. Each event can trigger additional events and call methods from the environment; which is a shared information structure. Once the stack is empty, metrics are computed or updated. The whole process, within a single run, repeats until the desired number of simulations, and the results confidence, is reached.

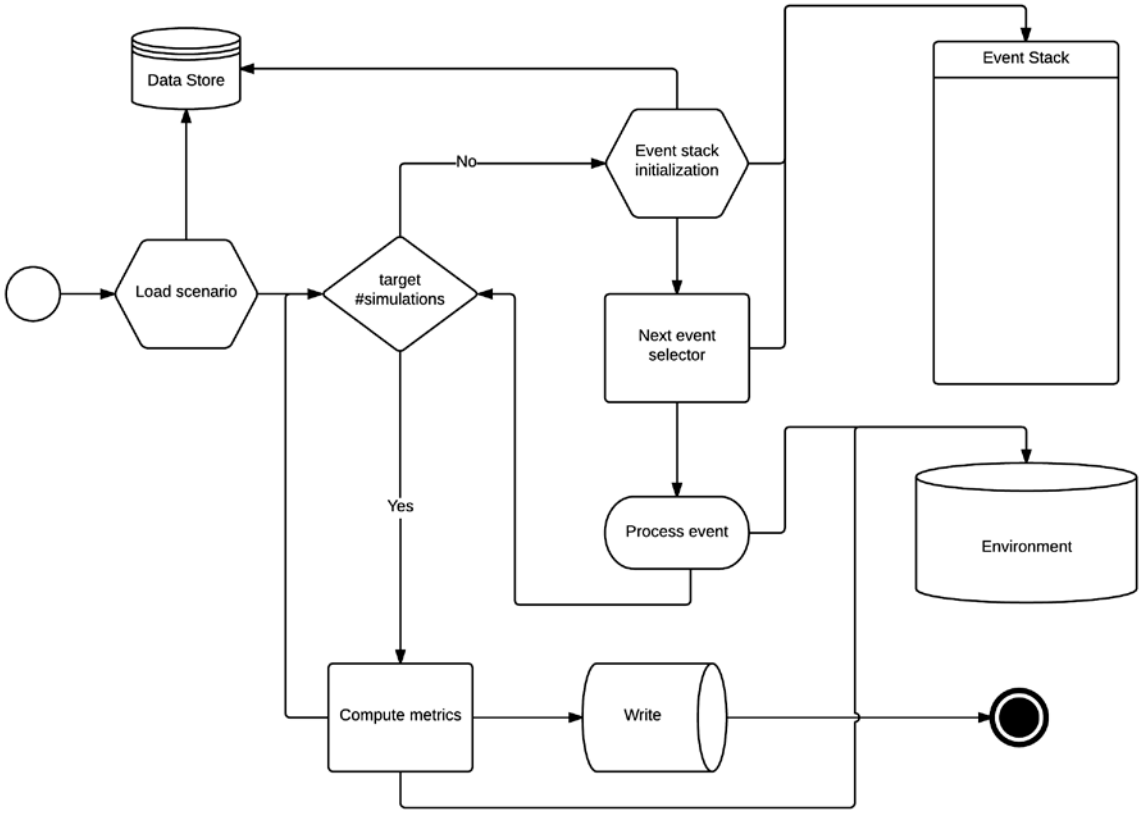


Figure 1. Main flow of a simulation

There is an important conceptual separation from the environment (e.g. agents) and the events (e.g. processes). The environment contains all of the agents and shared information of the model of a particular moment. Each actor is defined by a set of parameters (both static and dynamic) and provides a set of interface functions. Interface functions can be of two types: Soft functions, providing

partial answers or accepting imprecise, incomplete inputs; or hard functions, in which same-input returns the same-output.

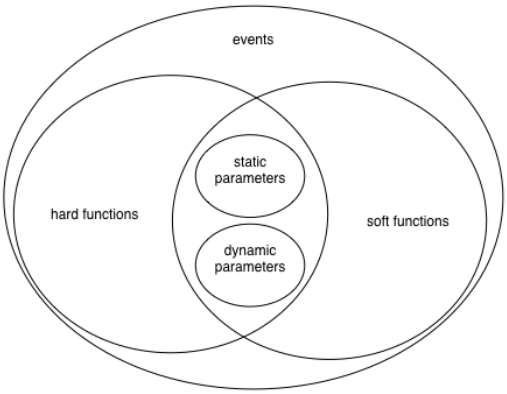


Figure 2. Languages hierarchy

On the other hand events are (sub)processes involving actors. Events are triggered in a certain moment of simulation and handled by the event stack. Depending on the current model status, the answer of the actors and the algorithm, new events can be queued or the environment can be modified (e.g. change the status of an actor).

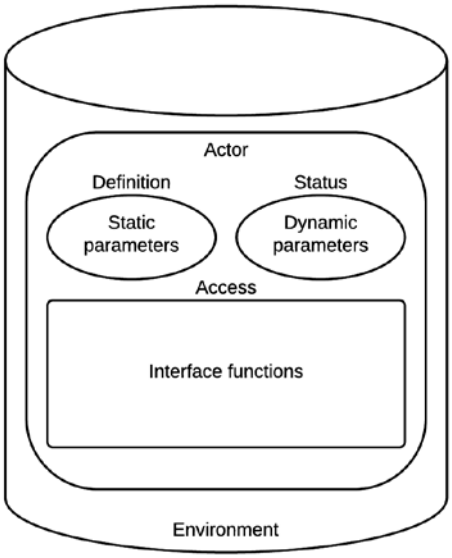


Figure 3. Environment definition

2.2 ComplexityCosts model implementation

A series of actors have been implemented for the ComplexityCosts model. The following is the full list and further definitions and details can be found in Section 3.

Table 1. Actors in the ComplexityCosts model

#	Name	Instances	Notes
1	Network Manager	0-1	Implementing Rule (EU N° 677/2011) lists at Article 4 the tasks to be performed by the Network Manager in pursuit of the functions. However, in the ComplexityCosts model the Network Manager tasks would be limited to balance demand and capacity profiles.
2	Air Navigation Services Provider	30-50	Manages the aircraft in flight or on the manoeuvring area of an airport, which is the legitimate holder of that responsibility.
3	Airport	150-500	In the ComplexityCosts model airport operations are limited to passenger boarding, aircraft off-block and in-block, taxi procedures, take-off and landing.
4	Airline Operator	100-2,000	Entities holding a valid AOC to operate commercial air transport. In the context of ComplexityCosts airlines are mainly restricted to flight operations and passenger transportation.
5	Flight	20,000-30,000	Operated by AOC holders and managed by ANSPs and the Network Manager.
6	Passengers	2,000,000-3,000,000	Full itineraries, individual flight tickets and fares are considered in the ComplexityCosts model.

The simulation flow is determined by the sequence of events. Each event, with the possible exception of initialization events, has a precursor event and may introduce this precursor event into the stack for further processing. The outcome and precursor events scheme for ComplexityCosts is as follows:

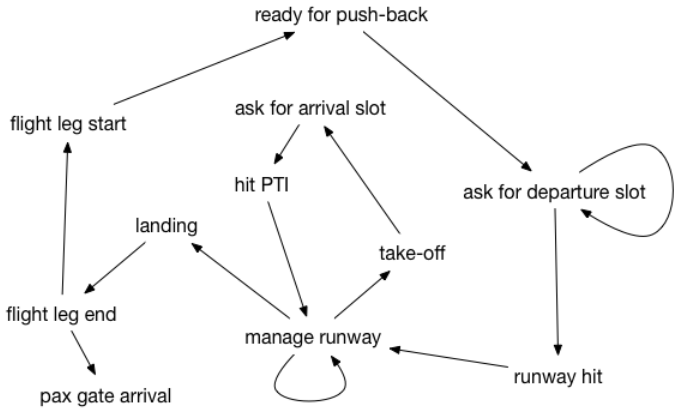


Figure 4. Sequence of events

From the flight perspective the most likely sequence of events is as follows: most flights initiate with a flight leg start event with time stamp given by the scheduled boarding time. Checks are performed in this event to decide whether the flight is ready for push-back or if it should be delayed (e.g. aircraft not ready, passengers are late, a regulation is in place, flight was cancelled, etc.). Once the flight is ready for push back a new event for departure slot is created to alert the departure manager that the flight is ready to off-block. In this case a taxi-out time is computed and a runway-hit event is introduced.

The departure airport selects arrival/departure sequence and therefore introduces a take-off or landing event. The former computes an estimated arrival time to the approach and introduces a prompt for arrival slot and hit PTI events, calling the manage runway event at destination airport. The

landing event calls for the flight leg end times after taxi and passenger gate arrival including flight connecting times.

Table 2. Events dependencies

Name	Triggering	Precursor(s)	Consequent(s)
ask for arrival slot	Airline	take-off,	hit PTI
ask for departure slot	Airline	ask for departure slot, ready for push-back	ask for departure slot, runway hit
flight leg end	Flight	landing,	flight leg start, pax gate arrival
flight leg start	Flight	flight leg end	ready for push-back,
landing	Flight	manage runway	flight leg end,
manage runway	Airport	runway hit, pti hit, manage runway	manage runway, take-off, landing
pax gate arrival	Pax	flight leg end	--
hit PTI	Flight	ask for arrival slot	manage runway,
ready for push-back	Airline	flight leg start, ready for push-back	ask for departure slot, ready for push-back
runway hit	Flight	ask for departure slot,	manage runway,
take-off	Flight	manage runway	ask for arrival slot

3 Framework Implementation

3.1 Overview of the software architecture

The ComplexityCosts software tool has been developed using MATLAB R2015a (8.5.0.197613) and follows the general file organization rules of the programming language. Each file containing code represents either a script or a single function, and no multiple functions can be declared in a single file. Functions and scripts are organized into folders. File names define function calls, which are case sensitive and cannot be repeated. These function calls must be unique.

Scripts are used for data preparation and initialisation only. The rest of the tool has been developed using functions. For example, entities are coded using special functions called constructors. Constructors are an abstraction from object-oriented programming defining the meta structure that can be afterwards instantiated (e.g. loaded in memory). There are two conceptually different classes of objects. One class defines the simulation features, code flow, memory management, processing, etc. The other class corresponds to the actual elements to be represented into the model (e.g. flights, passengers, airports, etc.)

Finally events are coded into regular functions using special headers to ensure there is enough information for the event to be handled and processed by the Event Stack. Events make use of methods contained in the previously defined objects (both classes) but never reversed. This distinction allows a more flexible software modification. Some events can also work in legacy mode, which uses older versions to ensure compatibility.

3.2 Input data preparation

Data used in the ComplexityCosts model come from many different sources in multiple formats. Prior to implementing the actual model a form of data audit is required. The data audit is composed of a series of algorithms aimed to improve the data set in one or more of the following facets: the data completion, consistency and relevancy.

To ensure the data is complete, each register in every database used is checked individually and ranked according to the amount of available information. When non-critical information is missing lines are tagged as *warning* and included in the model anyway but use placeholders as substitutes for the lack of information. If critical information is missing lines are skipped and tagged as *fatal error*. Examples of a data *warning* would be if a flight number is missing or when an aircraft configuration is not available. A *fatal error* would be missing a departure or arrival airport or a SOBT/SIBT.

Data consistency is checked individually against particular data fields, for instance the following code explores the consistency of flight legs (i.e. discarding legs in which the arrival and departure airport do not match).

Code 1. Flight leg data integrity check

```
n=length(fltNumList);
outgoing=zeros(n,1);
inbound=zeros(n,1);
registers=setdiff(unique(registrationList),'');
errors=0;
sequences=cell(1);
i=1;

for j=registers
  skip=false;

  unsorted=find(strcmp(registrationList,j));
  [~, I]=sort(SOBTsList(unsorted),'ascend');
  ind=unsorted(I);

  for k=1:(length(ind)-1)
      if strcmp(arrivalList{ind(k)}, departureList{ind(k+1)})
```

```
                sequences{i}=[sequences{i}, ind(k)];
            else
                if not(isempty(sequences{i}))
                    sequences{i}=[sequences{i}, ind(k+1)];
                    i=i+1;
                    sequences{i}=[];
                end
            end
        end

        if not(isempty(sequences{i}))
           sequences{i}=[sequences{i}, ind(k+1)];
           i=i+1;
           sequences{i}=[];
        end
    end
```

Regarding data relevancy some information was not available and it was estimated using existing data only. For instance the maximum airport capacity is estimated from the traffic sample searching for the maximum number of movements using a moving temporal window and a minimum value as follows:

Code 2. Airport capacity estimation from data

```
EU=find(strcmp(region,'ECAC'));
minOBT=min(SOBTsList);
maxIBT=max(SIBTsList);
width = 60/(24*60);
overlap = 15/(24*60);
capacity=NaN*ones(length(icao),1);

for i=EU;
    topmov=45;

    s=minOBT;
    t=s+width;

    Dep=find(strcmp(departureList, icao{i}));
    Arr=find(strcmp(arrivalList, icao{i}));

    while (s<=maxIBT)

        mov=0;

mov=mov+sum(((SOBTsList(Dep)+txo_mean(i)/(24*60))>=s)&((SOBTsList(Dep)+tx
o_mean(i)/(24*60))<t));

mov=mov+sum(((SIBTsList(Arr)-txi_mean(i)/(24*60))>=s)&((SIBTsList(Arr)-
txo_mean(i)/(24*60))<t));

        if (mov>topmov)
            topmov=mov;
        end

        s=s+overlap;
        t=s+width;
    end

    capacity(i)=ceil(topmov*0.982);
end
```

Data also required some transformation, for instance the current taxi model (in and out) uses a log-normal distribution, and therefore the parameters need to be estimated from the ones provided (i.e. sample average and standard deviation). This is done using the well-known statistics formula:

Code 3. Taxi parameters data transformation

```
function computeTaxiParameters

 n=length(data.name);

 data.taxi.out.sig=zeros(1,n);
 data.taxi.out.mu=zeros(1,n);
 data.taxi.in.sig=zeros(1,n);
 data.taxi.in.mu=zeros(1,n);


 for j=1:n

     if ( (data.taxi.out.std(j)~=0) && (data.taxi.out.std(j)~=0))

         M=data.taxi.out.mean(j)*parameters.simulation.timescale;
         S=data.taxi.out.std(j)*parameters.simulation.timescale;

         data.taxi.out.sig(j)=sqrt(log(((S^2)/(M^2)) + 1));
         data.taxi.out.mu(j)=log(M) -  ((data.taxi.out.sig(j)^2)/2);


     end

     if ( (data.taxi.in.std(j)~=0) && (data.taxi.in.std(j)~=0))

         M=data.taxi.in.mean(j)*parameters.simulation.timescale;
         S=data.taxi.in.std(j)*parameters.simulation.timescale;

         data.taxi.in.sig(j)=sqrt(log(((S^2)/(M^2)) + 1));
         data.taxi.in.mu(j)=log(M) -  ((data.taxi.in.sig(j)^2)/2);

     end
 end
```

Each of the data preparation script has to run once any of the data sources changes. They are all stored under the "preparation" directory.

3.3 Objects and classes definition

The ComplexityCosts software has been developed using object-oriented programming paradigm, therefore all variables are in fact objects (e.g. data structures containing data and functions). As with many of the class-based languages, MATLAB defines objects as typed classes. However, due to the particularities of the ComplexityCosts model, ad hoc constructor for objects has been developed. The main reason for this being the number of required objects, unused methods and computational power required.

All objects in the ComplexityCosts model inherit from the general class described below. This ensures that at least the defined methods are always available.

Code 4. General constructor header

```
function object = constructor()
```

```
        data={};

        object.meta=@meta;
        object.get=@get;
        object.set=@set;
        object.reset=@reset;
        object.add=@add;
        object.remove=@remove;

        object.oncall=@oncall;
        object.display=@display;
        object.backdoor=@backdoor;
        ...

    end
```

Simply calling the general constructor function inside of the individual object constructor then makes inheritance.

Code 5. Inheritance example

```
function exampleObj = objConstructor()
       exampleObj=constructor();
       ...
end
```

3.3.1 Meta-objects definition

Meta-objects are classes that are not part of the simulation, but rather constitute the (empty) simulator engine. Methods contained in those classes allow the program to execute and drive the simulation. This also captures the outputs and can be very useful when verifying and debugging the software.

The logConstructor defines open, write and close functions for two types of logs. Initialising sequences can be logged using the log.load methods, while the methods defined in log.exec would capture strings in execution only. Log is activated and deactivated using the binary global variable parameters.logging.enable.

Code 6. Log constructor header

```
function log = logConstructor()
     global parameters;

     floading=[];
     fexec=[];

     log=constructor();

     log.load.open=@openLoad;
     log.exec.open=@openExec;

     log.load.write=@writeLoading;
     log.exec.write=@writeExec;

     log.load.close=@closeLoad;
     log.exec.close=@closeExec;

       ...
end
```

The simulator class allows the simulation to start, stop and continue. Prior to the start of a simulation, all of the actors are reset to their initial status and output files are restarted. For the continue method, the actors do not restart nor do the event stack, and results are written carefully so no previous results are overwritten. The stop method simply halts the current simulation.

Code 7. Simulator constructor header

```
function driver = simulatorConstructor()

     global parameters eventStack airports airlines flights pax log
results ;

     driver = constructor();

     driver.simulate=@simulate;
     driver.continue=@cont;
     driver.stop=@stop;

     driver.restartModel=@restartModel;

       ...
end
```

The Event Stack manages the flow of the simulator by the sequence of events. Events can be added using the eventStack.add method, or deleted using the eventStack.delete method. The time stamp value of a given event can be read or modified using eventStack.setTime/getTime respectively. It is possible to explore the Event Stack using the auxiliary methods eventStack.display and eventStack.searchNext. Finally events are processed using the eventStack.executeNext method, which is the most important method of this class.

Code 8. Event Stack constructor header

```
function eventStack = eventstackConstructor()

     eventStack = constructor();

     events=cell(0);
     objects=cell(0);
     timeStamps=[];

     unprocessed=[];
     s=0;

     eventStack.display=@display;
     eventStack.isempty=@isempty;
     eventStack.add=@addEvent;
     eventStack.delete=@deleteEvent;
     eventStack.reset=@reset;
     eventStack.executeNext=@executeNextEvent;
     eventStack.searchNext=@searchNextEvent;
     eventStack.setTime=@setTimeEvent;
     eventStack.getTime=@getTimeEvent;

     eventStack.time=@time;

       ...
 end
```

The results manager gathers all of the necessary information to produce results metrics during execution. After, it stores the raw status of the model (e.g. actual flight plans and passenger itineraries) for each single run. It also computes the flight-centred and passenger-centred metrics and produces the final report of the simulation scenario. Additionally, the results manager implements debugging and calibrating methods to facilitate these tasks.

Code 9. Results constructor header

```
function results = resultsConstructor()

     global parameters flights airports pax airlines;

     results = constructor();

     results.createDirectories=@createDirectories;

     results.runsAnalysed=@runsAnalysed;
     results.saveResults=@saveResults;

     results.computeSingleRunMetrics=@computeSingleRunMetrics;

     results.writeFlightDelays=@writeFlightDelays;
     results.writePaxDelays=@writePaxDelays;

     results.writeAirportFlightsMetrics=@writeAirportFlightsMetrics;
     results.writeAirportPaxMetrics=@writeAirportPaxMetrics;

     results.writeCalibration=@writeCalibration;
     results.writeMetrics=@writeMetrics;

     results.estimatePdfs=@estimatePdfs;

     results.computeOriginalMetrics=@computeOriginalMetrics;
     results.writeMatlabSingleRun=@writeMatlabSingleRun;
     results.writeRawFlightsIndicator=@writeRawFlightsIndicator;
     results.writeRawPaxIndicator=@writeRawPaxIndicator;

       ...
end
```

3.3.2 Simulation-objects definition

Actors and the environment have been implemented using four simulation-objects defined by classes as follow. Each of them has a series of internal parameters under the structured variable data. In some cases the initial value is copied into a static variable called *backup* which is used when restarting the model. Each class has the same basic structure. First, the generic object constructor commences to create the basis object interface. Second, the class is initialized using values from the scenario definition and default parameters. Finally, the methods are defined. As per MATLAB standards, these method definitions are carried out in two steps: first function names are specified and then algorithms follow. For simplification only methods definitions are provided in this document. This may produce a general idea of the model and could allow further development to be carried out.

The airport basic methods contain some initializing procedures, such as computeTaxiParameters, createAirportConnectionMatrix, getNeighbourhood and computeCapacity. By default, the taxi parameters are estimated using a Log-Normal model. The airport connection matrix and the neighbourhoods are used to speed up computations when trying to re-accommodate passengers. Lastly, computeCapacity estimates maximum capacity from traffic data.

The methods getLastMov, getPTI, getAprox, getRunwayTime and occupyRunway are all related to the airport maximum throughput.The two methods getPTI and getAprox are used by the arrival manager to estimate arrival time, while getLastMov is used by the getRunwayTime to estimate the runway occupation time (ROT), while setting occupyRunway to true, so that each runway is occupied only once at a time.

The getMCT method returns the minimum connection time of the airport and the getConnectingFlights returns the gates for connecting passengers.

Code 10. Airports constructor header

```
function airports = airportsConstructor()

     global parameters flights pax;
     airports = constructor();

     data = loadCsvAirports(parameters.files.airports);

     computeTaxiParameters();

     airports.createAirportConnectionMatrix =
@createAirportConnectionMatrix;

     airports.getNeighbourhood=@getNeighbourhood;
     airports.computeCapacity=@computeCapacity;

     airports.getLastMov=@getLastMov;
     airports.getPTI=@getPTI;
     airports.getAprox=@getAprox;
     airports.getNextRunwayTime=@getRunwayTime;
     airports.occupyRunway=@occupyRunway;

     airports.getMCT=@getMCT;
     airports.getConnectingFlights=@getConnectingFlights;

       ...
```

Additionally there is a set of airport methods related to the taxi-in and taxi-out times:

Code 11. Airport constructor header (cont.)

```
     airports.getTaxiOutMean=@getTaxiOutMean;
     airports.getTaxiInMean=@getTaxiInMean;

     airports.getTaxiOutStd=@getTaxiOutStd;
     airports.getTaxiInStd=@getTaxiInStd;

     airports.computeTaxiParameters=@computeTaxiParameters;

     airports.computeTaxiOut=@computeTaxiOut;
     airports.computeTaxiIn=@computeTaxiIn;
```

Furthermore, there is a set of airport methods related to passengers; paxReady returns the number of passengers ready for boarding while waitingPax returns the list of passengers that need to be waited for or they will be stranded. There are also three additional methods to manage this two list: addWaitingPax, removeWaitingPax and iswaiting.

Code 12. Airport constructor header (cont.)

```
     airports.paxReady=@paxReady;
     airports.waitingPax=@waitingPax;

     airports.addWaitingPax=@addWaitingPax;
     airports.removeWaitingPax=@removeWaitingPax;
     airports.iswaiting=@iswaiting;
```

Finally there is a flight-related airport method that basically manages the departure and arrival queues. They are kept independent and the events prioritise the particular order for departures and

arrivals. For instance an event could ask the airport for the list of waiting passengers, and if it is too long then it could decide to prioritise an arrival over a departure or *vice versa*.

Code 13. Airport constructor header (cont.)

```
      airports.addToDepartureQueue=@addToDepartureQueue;
      airports.addToArrivalQueue=@addToArrivalQueue;

      airports.getDepartureQueue=@getDepartureQueue;
      airports.getArrivalQueue=@getArrivalQueue;

      airports.deleteFromQueue=@deleteFromQueue;
      airports.clearQueue=@clearQueue;
      airports.nonPaxQueued=@nonPaxQueued;
      airports.nextQueued=@nextQueued;
      airports.queueLength=@queueLength;
      airports.expectedQueuedTime=@expectedQueuedTime;

      airports.displayQueue=@displayQueue;

        ...
 end
```

The airline actor constructor contains the AOs cost models including passenger hard and soft costs and also non-passenger costs related to at-gate costs, extra taxi time, and on-route/arrival costs. Some of the functions are soft functions as they provide only an estimate for costs. It is only necessary to run the pre-computed functions once in order to save time between multiple model runs.

Code 14. Airlines constructor header

```
function airlines  = airlinesConstructor()

      global parameters airports pax flights;

      airlines = constructor();

      data = loadCscAlliances(parameters.files.airlines); % returns
data.alliance and data.partner

      data.hardPaxCost = loadCsvHardpaxCost(parameters.files.hardpaxcost,
data.type); % data.hardPaxCost.mean and .std

      data.softPaxCost = loadCsvSoftpaxCost(parameters.files.softpaxcost);

      data.nonPaxCost =
loadCsvNonpaxCost(parameters.files.nonpaxgate,parameters.files.nonpaxtaxi
, parameters.files.nonpaxroute,...
          parameters.files.nonpaxarrival );

      T=[0, 5, 10, 15, 30, 45, 60, 90, 120, 180,
240]/parameters.simulation.timescale;

      airlines.sameAlliance=@sameAlliance;

      airlines.softCostPerPax=@softCostPerPax;
      airlines.hardCostPerPax=@hardCostPerPax;
      airlines.overnightCost=@overnightCost;
      airlines.valueOfTimePerPax=@valueOfTimePerPax;

      airlines.precomputeArrivalCost=@precomputeArrivalCost;
      airlines.precomputeDepartureCost=@precomputeDepartureCost;
```

```
        airlines.costOfDelayDeparture=@costOfDelayDeparture;
        airlines.costOfDelayArrival=@costOfDelayArrival;

        airlines.aproxCostofDelayArrival=@aproxCostofDelayArrival;
        airlines.aproxCostofDelayDeparture=@aproxCostofDelayDeparture;

        airlines.totalNonpaxCost=@nonPaxCost;

        airlines.nonPaxGateCost=@nonPaxGateCost;
        airlines.nonPaxTaxiCost=@nonPaxTaxiCost;
        airlines.nonPaxRouteCost=@nonPaxRouteCost;
        airlines.nonPaxArrivalCost=@nonPaxArrivalCost;
        airlines.nonPaxCost=@nonPaxCost;

          ...
    end
```

The flight constructor contains a series of methods to enable identification, access and modification of parameters, such as maximum waiting time for passengers (for both premium and non-premium tickets), regulation status, cancellation, occupancy, turnaround time, wake category, etc. To see a more exhaustive list we refer the reader to D2.1.

Code 15. Flights constructor header

```
function flights = flightsConstructor()

    global parameters airports pax airlines;

    flights = constructor();

    if strcmp(parameters.files.source,'.mat')

        if not(exist([parameters.files.traffic,
parameters.files.source], 'file'))

            data = loadCsvTraffic([parameters.files.traffic,'.csv']);
            save(parameters.files.traffic, 'data');

        else
            load(parameters.files.traffic, 'data');
        end

    else
        data = loadCsvTraffic([parameters.files.traffic, '.csv']);
        save(parameters.files.traffic, 'data');
    end

    for ff=1:length(data.id)
        data.aircraft.mtt(ff)=data.aircraft.mtt(ff)*1.25;
    end

    data.maxWait=NaN*ones(1,length(data.id));
    data.maxInflexWait=NaN*ones(1,length(data.id));
    data.occupancy=zeros(1,length(data.id));
    data.regulated=zeros(1,length(data.id));
    data.cancelled=false*ones(1,length(data.id));

    resetTimes();

    flights.setMTT=@setMTT;
    flights.getMTT=@getMTT;

    flights.getPrevious=@getPrevious;
    flights.getNext=@getNext;
```

```
flights.isExcluded=@isExcluded;
flights.isPax=@isPax;

flights.getDepartureID=@getDepartureID;
flights.getArrivalID=@getArrivalID;
flights.getArrivalName=@getArrivalName;
flights.getRouteDuration=@getRouteDuration;
flights.longhaul=@longhaul;
flights.getAircraftType=@getAircraftType;
flights.getWake=@getWake;
flights.getRegistration=@getRegistration;
flights.getAOType=@getAOType;
flights.getAO=@getAO;
  ...
```

Of greater interest are the following methods, which enable flight regulations (e.g. imposing a delay before departure) and even the ability to completely cancel a flight and therefore try to re-accommodate all the passengers into other flights.

Code 16. Flights constructor header (cont.)

```
  ...
flights.cancelled=@cancelled;
flights.cancel=@cancel;
flights.distributeCancellations=@distributeCancellations;

flights.addRegulation=@addRegulation;
flights.regulated=@regulated;
  ...
```

Each flight has multiple time stamps for the each phase, and the following methods allow the user to read and modify those values. Updating calculated and estimated times have a cascade effect; subsequent times will also be estimated or calculated (e.g. if ETOT is modified, then also ETA and EIBT are modified).

Code 17. Flights constructor header (cont.)

```
  ...
flights.updateEOBT=@updateEOBT;
flights.updateCOBT=@updateCOBT;
flights.updateCTOT=@updateCTOT;
flights.updateCTA=@updateCTA;
flights.updateCIBT=@updateCIBT;

flights.setART=@setART;
flights.setPR=@setPR;
flights.setAOBT=@setAOBT;
flights.setARWR=@setARWR;
flights.setATOT=@setATOT;
flights.setPTI=@setPTI;
flights.setATA=@setATA;
flights.setAIBT=@setAIBT;

flights.getART=@getART;
flights.getPR=@getPR;
flights.getAOBT=@getAOBT;
```

```
            flights.getARWR=@getARWR;
            flights.getATOT=@getATOT;
            flights.getPTI=@getPTI;
            flights.getATA=@getATA;
            flights.getAIBT=@getAIBT;

            flights.getCOBT=@getCOBT;
            flights.getCTOT=@getCTOT;
            flights.getCTA=@getCTA;
            flights.getCIBT=@getCIBT;

            flights.getEOBT=@getEOBT;
            flights.getETOT=@getETOT;
            flights.getETA=@getETA;
            flights.getEIBT=@getEIBT;
              ...
```

Finally, there is a series of passenger-related airport functions. These functions allow the program to associate passengers to flights and change the status/location of certain passengers depending on their flight status. It is also possible to add new passengers or to remove passengers (e.g. departing before actually waiting). The removed passengers may be re-accommodated or added to a waiting list.

Code 18. Flights constructor header (ends)

```
            flights.addPax=@addPax;
            flights.removePax=@removePax;

            flights.computeOccupancy=@computeOccupancy;
            flights.freeSeats=@freeSeats;
            flights.occupancy=@occupancy;
            flights.totalSeats=@totalSeats;

            flights.computePaxWaitingThresholds=@computePaxWaitingThresholds;
            flights.getPaxWaitingThresholds=@getPaxWaitingThresholds;

            flights.paxListBackup=@paxListBackup;

            flights.getPaxList=@getPaxList;
            flights.getOriginalPaxList=@getOriginalPaxList;
            flights.searchFlights=@searchFlights;
            flights.searchConnection=@searchConnection;

            flights.computeTicketRange=@computeTicketRange;
            flights.getTicketTange=@getTicketTange;
              ...
        end
```

The passenger-class constructor parameters are stored in a structured variable called data. It is first loaded as defined by the scenario parameters. Some passengers are flagged according to the outcomes of the simulation. For example, waitingReady signifies that the passengers are ready to be re-accommodated, while broken means that the passenger has missed a connection, and excluded means that they are excluded from the passenger metrics (e.g. not enough data to produce reliable results). Passengers are added to flights using the addPaxToFlights which needs to be run only once per simulation. Also previousFlight and nextFlight indicate the next flight in the passenger sequence of flights. The ending and starting methods indicate the O/D pairs, while the singleton method returns true when passengers take a direct flight. The ticket type is defined by the methods isFlex (i.e. premium) and isInflex (i.e. non-premium).

Code 19. Passenger constructor header

```
function pax = paxConstructor()

     global parameters flights airports log airlines;

     pax = constructor();

     if strcmp(parameters.files.source,'.mat')

         if not(exist([parameters.files.pax, parameters.files.source],
'file'))

              data  = loadCsvPax([parameters.files.pax,'.csv']);
              save(parameters.files.pax, 'data');

         else
             load(parameters.files.pax, 'data');
         end

     else
         data  =
loadCsvPax([parameters.files.pax,parameters.files.source]);
         save(parameters.files.pax, 'data');
     end

     data.number=zeros(1,length(data.name));
     data.nFare=cell(1,length(data.name));
     data.waitingReady=NaN*ones(1,length(data.name));
     data.broken=zeros(1,length(data.name));
     data.excluded=false*ones(1,length(data.name));

     backup=data;

     pax.addPaxToFligths=@addPaxToFligths;
     pax.isExcluded=@isExcluded;

     pax.nextFlight=@nextFlight;
     pax.previousFlight=@previousFlight;

     pax.ending=@ending;
     pax.starting=@starting;

     pax.singeltons=@singeltons;
     pax.broken=@broken;

     pax.getRoute=@getRoute;
     pax.getNumber=@getNumber;
     pax.getFares=@getFare;
     pax.getCategory=@getCategory;

     pax.isFlex=@isFlex;
     pax.isInflex=@isInflex;

       ...
```

Finally, there is a set of methods implemented to allow the model to re-accommodate passengers when connections are missed. Note that some functions are in fact soft functions, for instance searchReaccommodation is a soft function since it searches for future accommodation under the current status of the system. On the other hand, accommodate is a hard function as it actually assigns a passenger to a flight.

Code 20. Passenger constructor header (end)

```
        ...
    pax.searchReaccommodation=@searchReaccommodation;
    pax.searchNextDayFlight=@searchNextDayFlight;
    pax.reaccommodate=@reaccommodate;
    pax.checkReaccomodation=@checkReaccomodation;

    pax.linkedPax=@linkedPax;
    pax.reaccommodatedFrom=@reaccommodatedFrom;
    pax.reaccommodated=@reaccommodated;

    pax.missConnection=@missConnection;
    pax.newRoute=@newRoute;

    pax.setArrivalTime=@setArrivalTime;
    pax.getArrivalTime=@getArrivalTime;
    pax.computeArrivalTime=@computeArrivalTime;
    pax.waitingReady=@waitingReady;
    pax.setWaitingReady=@setWaitingReady;

    pax.getEstArrivalTime=@getEstArrivalTime;
        ...
end
```

3.4 Events implementation

All ComplexityCosts tool events are stored in the \events folder, which contains the eight implemented algorithms. Each event has a common declaration structure. First the name of the event is given by the function's call and three parameters: the actor triggering the event, the time in which the event should be processed and the event stack to which it belongs. It would be possible to have more than one event stack in future improvements of the tool. New events can be added to the stack using the event stack method add followed by function identifier and actor and time parameters.

Code 21. Event basic definition

```
function eventStack = event_name(actor, time, eventStack)

    global parameters log results airports flights pax airline;

    % event code

    eventStack.add('@new_event', new_actor, new_time);

end
```

In order to get the simulation started some events need to be introduced into the stack, this is done using the simulator driver method restartModel. It will, among other things, add to the stack all of the leg flight start events without a previous connecting flight.

Code 22. Events initialisation

```
driver.restartModel=@restartModel;

function restartModel()
         log.reset();
         results.reset();
         airlines.reset();
         airports.reset();
         flights.reset();
         pax.reset();
         eventStack.reset();
         initEvents();
         l='=======================Model
restarted============================';
         log.exec.write(l);
end

function initEvents()
       n=flights.number();
     J=1:n;
     for j=1:length(J)
      r=randi(length(J));
         i=J(r);
         J(r)=[];
         if (flights.getPrevious(i)==0)
               g= (rand<0.95)*random('exp', 15/1440);
             while (g>(60/1440))
               g= (rand<0.95)*random('exp', 15/1440);
             end
             eventStack.add('@flightLegStart', i, flights.getEOBT(i) -
g);
         end
     end
 end
```

Example of a take-off event:

Code 23. Take-off event

```
function eventStack = takeoff(ind, time, eventStack)
     global parameters airports flights pax log results;

     flights.setATOT(ind, time);

     departureDelay=flights.getATOT(ind)-flights.getETOT(ind);

     recover=(1+(departureDelay>5/1440))*departureDelay-(5/1440);

     recover=min(recover,
3*parameters.route.std*flights.getRouteDuration(ind));

     recover=max(recover, -
2*parameters.route.std*flights.getRouteDuration(ind));

     h=parameters.route.std*flights.getRouteDuration(ind)*randn -
recover;

     realDuration = flights.getRouteDuration(ind) + h;

     aprox = time + max(realDuration -
airports.getAprox(flights.getArrivalID(ind)),0);

     eventStack.add('@askForArrivalSlot',ind, aprox);
```

```
    if parameters.logging.enable
        s=sprintf('(%s) %s:%d taking-off from %s:%d, expected to reach
%s:%d PTI in %s, queued: %s, acc. delay: %s', ...
            datestr(time),  flights.getID(ind),ind,...

airports.getName(flights.getDepartureID(ind)),flights.getDepartureID(ind)
, ...

airports.getName(flights.getArrivalID(ind)),flights.getArrivalID(ind)
,...
            datestrmin(flights.getRouteDuration(ind)-
airports.getPTI(flights.getArrivalID(ind))),...
            datestrmin(time-flights.getARWR(ind)), datestrmin(time-
flights.getETOT(ind)));
         log.exec.write(s);
    end

 end
```

As reviewed there are some dependencies between the events and the actors involved, and the following table summarizes those dependencies. If an existing method or any actor is modified, the corresponding events would be revised and tested accordingly.

Table 3. Events and actors dependencies

name	trigged by	airports	airlines	flights	passengers
leg flight start	init			✓	✓
ask for departure slot	flight	✓	✓	✓	
manage runway	airport	✓			
take off	flight	✓		✓	
hit PTI	flight			✓	
ask for arrival slot	flight	✓		✓	
pax gate arrival	passenger	✓			✓
flight leg end	flight			✓	

4 Modules integration

4.1 Passenger and traffic model

As introduced in Deliverable 1.3, ComplexityCosts is building new 2014 passenger itineraries from an existing 2010 dataset, developed in-house using passenger data sourced from IATA's PaxIS dataset assigned to individual flights supplied by EUROCONTROL's PRISME data service. The requirements for this new dataset include:

- Credible passenger itineraries, consisting of single and multiple flight legs, with ticket price and simple seat class distinction (premium/non-premium);
- Achievable connections for passengers with multiple flight legs, based on schedule times and MCTs;
- Respecting (target) load factors per route and the maximum seating capacity per aircraft;
- Closely matching the target number of passengers per flight (stochastically distributed*);
- Total passengers and overall average load factor should be consistent with a busy Friday in September.

** Each flight will have a stochastically distributed target number of passengers, taking account of the increase (or decrease) in passengers on the route.*

With the programming assistance of Adeline de Montlaur from the Universitat Politècnica de Catalunya (a visiting researcher at UoW), considerable progress has been made with the first stage of building the 2014 passenger itineraries: assigning existing 2010 itineraries onto 2014 flights. In summary:

1. **Assign three flight leg itineraries:**
    a. Assign the premium passengers with the following criteria: first assign an itinerary where the airline matches the existing one for the longest leg. To do so the following "cost" function is minimised (where "L1" is the first flight leg in a multi-flight itinerary, and so on):

    AO_Cost = distance L1 × operator L1 + distance L2 × operator L2 + distance L3 × operator L3, with operator L*i* = 1 for same airline, 2 for same partner, 3 for same alliance.

    If several itineraries have the same "cost" then minimise the total travel time.

    b. Next assign the non-premium passengers with the following criteria: first assign an itinerary where the airline matches the existing one for the longest leg (same process as 1a).

    If several itineraries have the same "cost" then assign randomly.

2. **Assign two flight leg itineraries:**
    a. Same process as 1a.
    b. Same process as 1b.
    c. Assign two flight leg itineraries found with different airlines to the originally assigned airlines.

3. **Assign single flight itineraries:**
    a. First assign premium passenger prioritising the same airline (then partner, then alliance) from the existing itinerary.
    b. Next assign non-premium passenger prioritising the same airline (then partner, then alliance) from the existing itinerary.
    c. Assign single flight itineraries found with different airlines to the originally assigned airlines.

Testing shows the first stage is able to reassign a high percentage of existing 2010 itineraries onto 2014 flights. Final first stage assignment awaits further flight data cleaning and enhancement, such as allocating the maximum seating capacity to each passenger aircraft and scheduled departure/arrival times to each passenger flight in scope.

The second stage will assign additional passengers onto flights that have not reached their target number of passengers. Where possible, these will be drawn from existing available itineraries unassigned during the first stage (e.g. by re-routing unassigned connecting passengers via an alternative hub) and from new GDS data (anonymised September 2014 sample supplied by a large GDS).

The third stage will assign passengers onto flights on new routes, i.e. routes that do not exist in the 2010 dataset. Again where possible, these will be drawn from existing itineraries unassigned during the first and second stages and from new GDS data.

For flights without passengers, an estimation of the expected number of passengers on the route per day will be made using monthly Eurostat O/D data and available seating capacity per route. Where available, route load factors and other specific per-route information will inform the process (e.g. O/D statistics published by civil aviation authorities).

4.2 Mechanism model

4.2.1 Increasing ATCO hours in selected sectors

As presented in Section 4.3 (Disturbance models) a shortage of ATCO hours in a sector is translated into delay due to an ATFM regulation due to staff issues or in some cases due to capacity issues, as a configuration with more sectors is not possible to be used. Therefore, increasing the ATCO hours in selected sectors could be modelled as an increment in the capacity of the ACC during the period of time of the disruption leading to a reduction on the impact of that disruption (i.e., more capacity and therefore less delay assigned) and/or a reduction of the duration of the ATFM regulation.

The ACC that will benefit from this strategy will be selected based on the areas which generated more delay due to ATCO staff issues and that might benefit the most from this mechanism. These areas will be selected based on the analysis of 2014 ATFM regulations as recorded in the DDR2 dataset (within the period AIRACs 1313 to 1413 (i.e., from the 12th December 2013 until the 07th January 2015)). Delay directly assigned to flights by regulations that were issued due to staff shortage will be considered, identifying, in this manner, the ACCs with more number of regulations and delays. ATCOs availability will be extracted from DDR2 for the different ACCs during the same period of time, identifying the areas where a shortage of ATCO staff led to regulations marked as staff, capacity or ATC routing. ACCs that will benefit of a different rostering, and hence of more ATCO hours, will present a higher gap between the maximum number of ATCOs available in the vicinity of the time of the regulations and the available during the regulations; ACCs where this difference is small but that still issued regulations due to staff shortage might need new sectorisations. More information regarding to the analysis of the ATFM regulations due to staff can be found in Section 4.3; the spatial scope of staff capacity disruption will define the areas that will benefit from this mechanism.

Values from the ATM Cost-effectiveness 2013 Benchmarking Report (EUROCONTROL, 2015a) will be used to estimate the cost of extra hours of ATCOs at the selected ACCs where the mechanism will be implemented. Parameters such as the cost per composite flight-hour will also be considered to adjust the tactical cost of managing the traffic. Other costs such as support costs should also be considered at the tactical level.

Finally, from other sources such as (NATS, 2015) or (SENASA, 2015) information on the training cost for ATCOs will be estimated and incorporated as strategic costs of the mechanism.

4.2.2 Dynamic cost indexing

When modelling the benefit of dynamic cost indexing mechanism, the flights of airlines implementing this strategy will assess the cost of their flights at different speeds at different times during their operations (e.g. before take-off, at top of climb and during the cruise) and select the speed which minimises the cost for the airline considering fuel and delay costs. This mechanism will be based, in part, on the work developed in the project CASSIOPEIA (SESAR, 2013) and its extension DCI-4HD2D (SESAR, 2014).

The costs of implementing this mechanism include at a strategic level the systems that need to be in place in the aircraft (e.g. electronic flight bags) and in the airline operation centre (AOC) to compute the cost index to be used for a particular flight, and the required training to use such a system. Tactically, the cost of the communication between the pilot and the AOC should be considered.

In light of the difficulties of obtaining such costs, we have contacted the only known provider of advanced dynamic cost-indexing solutions, in the European market. Updates will be provided in subsequent reporting. The project team has made assurances to the supplier regarding disclosure, such that it is not identified in this report, pending related permissions. If necessary, only anonymised cost data would be used in future reporting.

4.2.3 A-CDM and improved airline passenger reaccommodation policies

At the airports where A-CDM is deployed, higher predictability is expected on the operations, as information is shared among the different stakeholders. Therefore, A-CDM will be modelled as an improvement on the minimum turnaround time required per aircraft. This means that the reactionary delay generated at an airport might be reduced. For each airport operation, a minimum time required to have the aircraft ready for the next leg can be estimated - these estimations will be reduced at airports with A-CDM. Another benefit that will be incorporated into the modelling is a better estimation of the taxi-in and taxi-out times reducing the delay generated at the airports.

The cost of the A-CDM mechanism will be estimated from different sources including different cost benefit analyses (EUROCONTROL, 2008; Deutsche Flugsicherung, 2010; Deutsche Flugsicherung, 2013). These references will allow us to allocate the cost to the different stakeholders at the airport. The same sources will be used to quantify the benefits on airport operations (i.e. on the predictability of the traffic and the turnaround times). The main reference for the implementation of A-CDM in Europe is EUROCONTROL (2015b), this could be used to model the stakeholders' uptake.

The passenger reaccommodation policies could be implemented as decision algorithms for flights with connecting passengers. These policies will include passenger wait/no-wait rules and enhanced policies to reaccomodate passengers that have missed their connection.

Due to the difficulty of obtaining passenger disruption management software costs, we have contacted the four major suppliers of such solutions in the European market. Updates will be provided in subsequent reporting. The project team has made assurances to the suppliers regarding disclosure, such that these suppliers are not identified in this report, pending related permissions. If necessary, only anonymised cost data, from such sources, would be used in future ComplexityCosts reporting.

4.3 Disturbance models

As presented in Figure 5, different types of disruptions will be modelled: Industrial actions, staff capacity restrictions, airport weather restrictions and background ATFM delays. The three former will require to model their scope (geographical and temporal) and their impact, the later will be modelled based on the ATFM delay observed on the baseline day (see section on Background ATFM for more details).

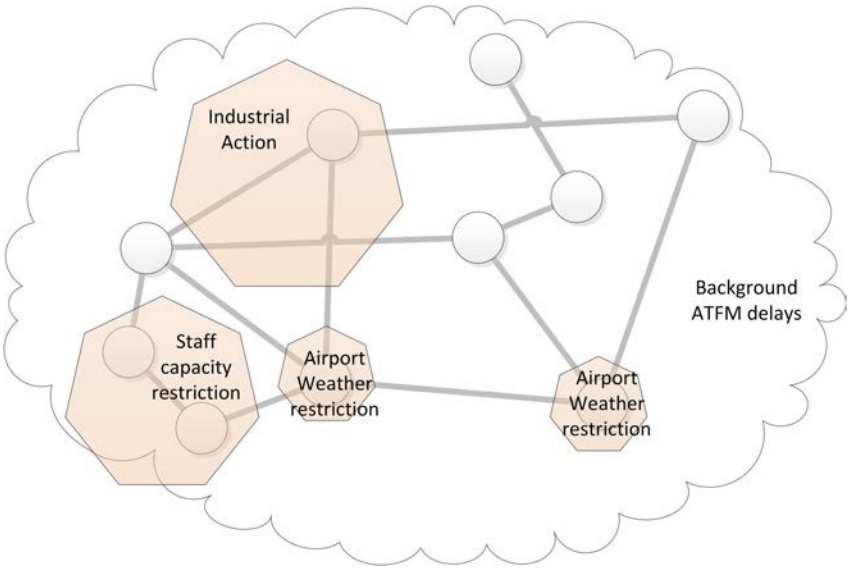


Figure 5. Diagram of the disruptions that will be modelled.

4.3.1 Background ATFM

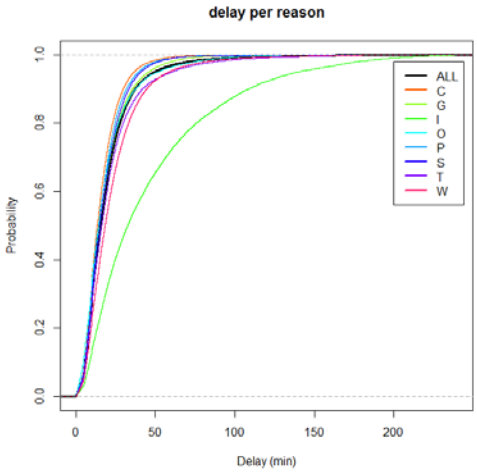
In ComplexityCosts, the cost benefit obtained when applying a given mechanism will be assessed based on the mitigation obtained for different disruptions. Therefore, specific disruptions will be modelled including their scope (spatial and temporal) and their impact on the traffic. In order to realistically model the traffic operations in Europe, however, the modelling of background ATFM delay in all other areas where the specific disruption/mechanism are not analysed should be included in the model. For this purpose, modelling the background ATFM delay and the effect of the disruptions in terms of ATFM delay, the ATFM delay directly generated by all the regulations implemented in Europe during the AIRACs 1313 to 1413 (i.e., from the 12th December 2013 until the 07th January 2015) has been analysed. The ATFM regulations are categorised based on the main reason that triggered them as indicated in Table 4.

Table 4. ATFM regulation causes

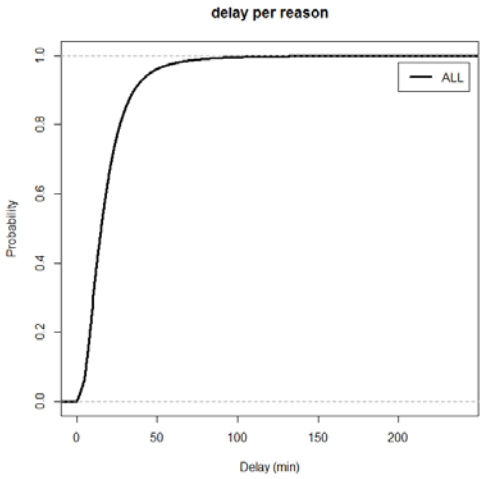
ATFM regulation cause code	ATFM regulation cause
C	ATC Capacity
G	Aerodrome Capacity
W	Weather
T	Equipment (ATC)
E	Equipment (Non ATC)
I	Industrial action
M	Military activity

R	ATC routing
S	ATC staffing
V	Environmental issues
D	De-icing
P	Special event
O	Other reason
U	Unknown reason

Figure 6(a) presents the cumulative probability distribution of the positive delay by main regulation reasons (reasons which regulations where the most penalising regulation directly assigning delay to 10,000 flights or more during the analysed period are showed). Industrial actions assign higher delays than the rest of the regulations, and as industrial actions are one of the particular disruptions that will be analysed in ComplexityCosts, these regulations can be removed from the modelling of the *nominal* background ATFM delay leading to the cumulative distribution of delay presented in Figure 6(b).



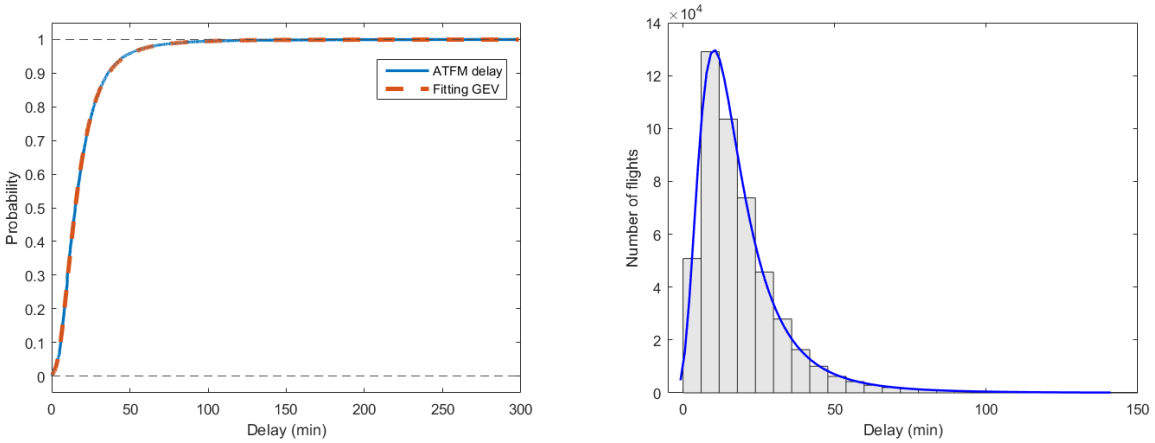
(a) Reasons with at least 10,000 flights with delay assigned.



(b) All reasons with at least 10,000 flights with delay assigned without industrial actions.

Figure 6. Cumulative probability distribution of ATFM delay per reason.

If only positive assigned delays up to 300 minutes are analysed (99.9% of the total number of flights which got delay assigned which represents 99.8% of the total ATFM delay), the ATFM delay probability distribution can be approximated by a Generalised Extreme Value distribution with parameters μ=11.93, σ=8.31 and ξ=0.23 (with a significance of p=0.0769), see Figure 7. Other distributions might be considered for this fitting.



(a) Accumulative probability distribution (b) Histogram

Figure 7. ATFM delay without industrial actions and with fitting

The previously described distribution could be used to generate delay to assign to the modelled flights. The traffic in ComplexityCosts is based on the operations of the 12th September 2014 and willing to model the operations as realistic as possible, the delay should not be assigned completely randomly to the flights. Flights going through regulations should have higher probability to get delay assigned and the introduction of reactionary delay propagation due to the assignment of high background ATFM delay to flights that in the original data had a small or null delay should be avoided. Moreover, the background ATFM delay should follow the distribution of the delay that was originally present in the operations of the 12th September 2014; as shown in Figure 8, the distribution of delay for the day under study sits in a middle position with respect to the delay distribution of the closest days that were short listed as *nominal* day and it is similar to the distribution of the delay presented previously with the analysis of the 1313 to 1413 AIRACs.

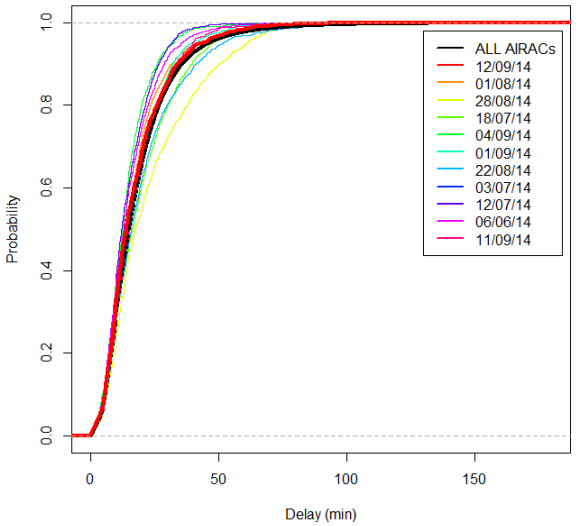


Figure 8. Cumulative distribution of delay for all delay, for selected day and others

To ensure that we meet the previously described restrictions, the suggested methodology to assign delay to flights adding some degree of randomness to the results but maintaining the principles of the delay of the 12th September 2014 is as depicted in Figure 9. This delay analysis is an *original* analysis from DDR2 data not extracted from a reference and it consists of the following steps:

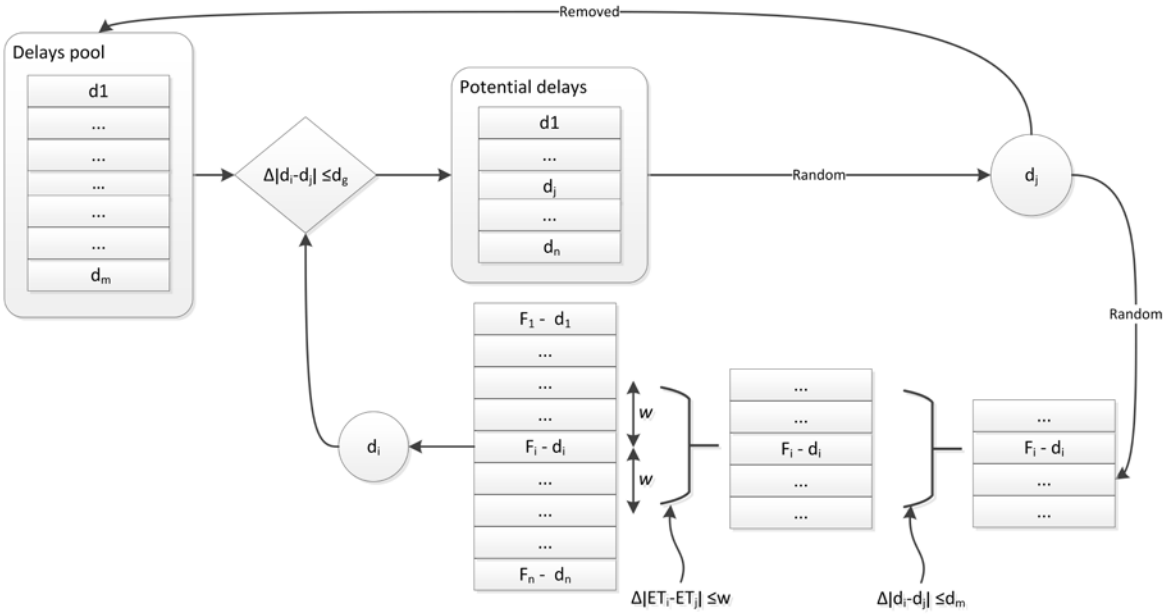


Figure 9. Delay generation concept

First a pool of potential delays is generated. This pool of delays contains all the ATFM delays that were generated on the 12th September 2014. Then in order to maintain the delay on the flights that go through traffic volumes that were regulated, the idea is that the delay is assigned to flights that operated in similar conditions.

1. Delay generation

   A delay is selected for each flight which had ATFM delay assigned in the original data. The delays that are within a given delay window (d_g) from the flight original delay (d_i) will be selected from the pool of delays. A delay will be randomly selected (d_j) among them and withdraw from the pool of delays. The delay which triggers the selection and the delay selected will be randomly chosen. This ensures that a given regulation will generate a total delay similar to the one in the original data, but not necessarily the same, adding some randomness; and as the selected delay is removed from the pool of delays, the distribution of delays over the flights during the day is maintained as in the original data. The range used to select the flights (d_g) will allow us to control the *uncertainty level* in terms of where the delay is generated.

2. Delay assignment

   Once the delay has been selected, the next step is to select which flight will get the delay assigned. The set of flights that enter the traffic volume around the same time as the flight which generated the delay are selected (i.e., flights with an entry time in the traffic volume within a given number of minutes (*w*)); in order to ensure that reactionary delay is not generated due to this background ATFM delay assignment, only the flights which originally had a delay with a difference smaller than a given threshold (d_m) with respect to the flight which generated the delay will be kept as potential flights to get the delay assigned. These thresholds will be also used to control the desired degree of variability with respect to the initial data.

The following example illustrates this two-step approach. Figure 10 presents the location of the traffic volume LSGL4W where regulation GLW412 was implemented between 7h30 and 10h30 on the 12th of September 2014. Table 5 shows some of the flights that entered the regulation and the ATFM

delay that they got assigned. Using an d_g of 15 minutes the delay is selected from the pool of delay as previously explained.

In Table 5, the flights within the window of 30 minutes are presented (w=30 min), from those only the ones that have a difference in their original ATFM delay of d_m=±15 minutes are kept as possible flights where the newly generated delay will be assigned. This random selection is presented in the *Flight selected* column in the table; and finally, the delay assigned is shown.

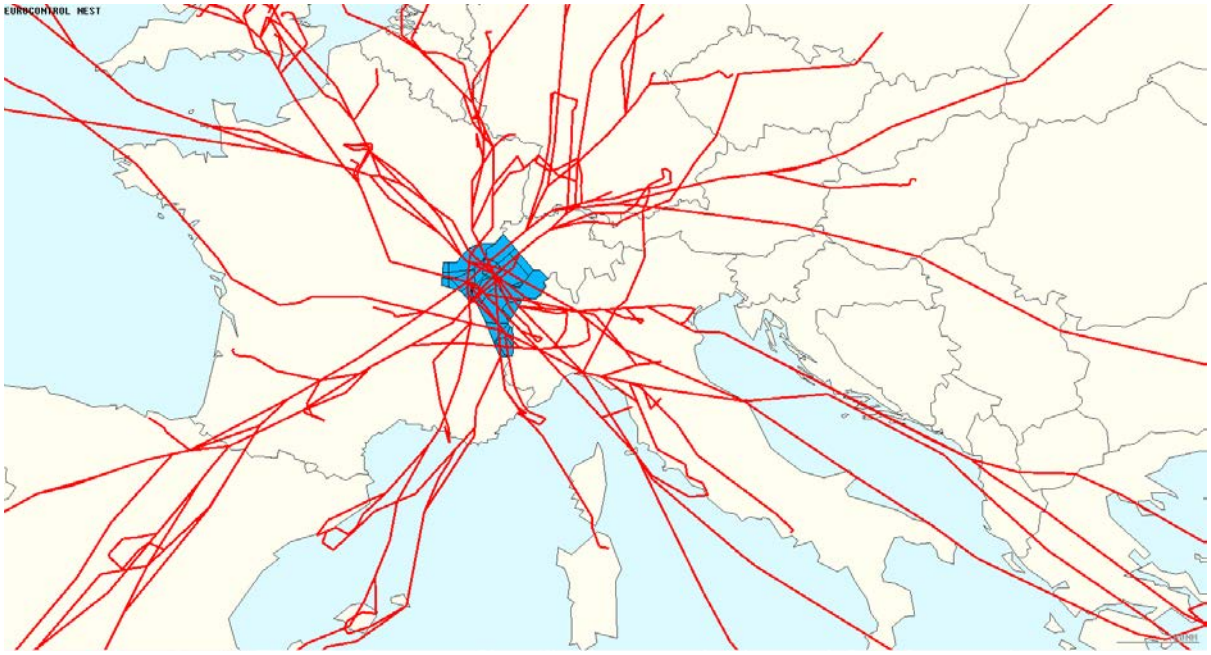


Figure 10. Traffic flying over LSGL4W where Regulation GLW412 was implemented.

Table 5. Regulation assignment example

Flight ID	Entry time in TV	Original ATFM Delay	Delay selected from pool of delay with d_g=15 min	Possible flights t=30min	Possible flights t=30min with ΔOriginal ATFM ≤ 15 min	Flight selected	Delay assigned
F1	08:48:05	0	0				0
F2	08:48:16	0	0				0
F3	08:48:34	0	0				0
F4	08:52:39	0	0				23
F5	08:57:37	0	0				0
F6	09:00:06	8	12	F1 ... F30	F1 ... F8, F10 ... F18, F20...F30	F25	0
F7	09:01:04	0	0				8
F8	09:02:13	0	0				13

Flight ID	Entry time in TV	Original ATFM Delay	Delay selected from pool of delay with d_g=15 min	Possible flights t=30min	Possible flights t=30min with ΔOriginal ATFM ≤ 15 min	Flight selected	Delay assigned
F9	09:02:35	40	26	F1 ... F30	F9, F19	F9	26
F10	09:03:25	0	0				0
F11	09:03:28	0	0				6
F12	09:03:44	0	0				0
F13	09:04:48	0	0				0
F14	09:05:22	14	12	F1 ... F31	F1 ... F8, F10 ... F18, F20 ... F31	F28	9
F15	09:06:07	10	23	F1 ... F31	F1 ... F8, F10 ... F18, F20 ... F31	F4	24
F16	09:06:21	0	0				6
F17	09:06:42	5	11	F1 ... F31	F1 ... F8, F10 ... F18, F20 .. .F31	F29	8
F18	09:07:30	16	9	F1 ... F31	F6, F14, F15, F17, F18, F22 ... F31	F14	21
F19	09:07:50	44	30	F1 ... F31	F9, F19	F19	30
F20	09:08:46	19	21	F1 ... F32	F6, F14, F15, F17, F18, F20, F22 ... F31	F18	10
F21	09:09:15	0	0				0
F22	09:11:15	8	10	F1 ... F33	F1 ... F8, F10 ... F18, F20 ... F32	F20	5
F23	09:11:44	8	8	F1 ... F33	F1 ... F8, F10 ... F18, F20 ... F32	F7	0
F24	09:12:30	18	13	F1 ... F35	F6, F14, F15, F17, F18, F20, F22 ... F31, F33	F24	13
F25	09:13:04	9	13	F1 ... F35	F1 ... F8, F10 ... F18, F20 ... F32, F34, F35	F8	12
F26	09:13:47	10	11	F1 ... F35	F1 ... F8, F10 ... F18, F20 ... F32, F34, F35	F26	11
F27	09:14:31	13	24	F1 ... F36ACC	F1 ... F8, F10 ... F18, F20 ... F32, F34 ... F36	F15	0

Flight ID	Entry time in TV	Original ATFM Delay	Delay selected from pool of delay with d_g=15 min	Possible flights t=30min	Possible flights t=30min with ΔOriginal ATFM ≤ 15 min	Flight selected	Delay assigned
F28	09:20:31	9	6	F4 ... F36	F4 ... F8, F10 ... F18, F20 ... F32, F34 ... F36	F16	12
F29	09:25:00	7	6	F5 ... F36	F5... F8, F10 ... F18, F20 ... F32, F34 ... F36	F11	11
F30	09:28:15	6	12	F6 ... F36	F6... F8, F10 ... F18, F20 ... F32, F34 ... F36	F35	0
F31	09:35:33	13	8	F14 ... F36	F14 ... F18, F20 ... F32, F34 ... F36	F17	0
F32	09:38:19	0	0				0
F33	09:40:22	29	21	F21 ... F36	F24, F33	F33	21
F34	09:42:12	2	5	F21 ... F36	F21 ... F23, F25 ... F32, F34 ... F36	F22	0
F35	09:42:38	0	0				12
F36	09:44:03	0	0				0

4.3.2 Staff capacity

Staff capacity regulations are one of the disruptions that will be explicitly modelled in ComplexityCosts. In this case, there is a need to model the spatial and temporal scope of the regulation and its intensity. Figure 11 shows the ACCs that reported any ATFM regulation categorised as *ATC staff* issues during the period AIRAC 1313 to AIRAC 1413. As shown, there are a significant number of ACCs (50), which reported regulations due to staff. In ComplexityCosts, we are interested in the ACCs, which had a higher impact on the delay due to those regulations and on those which could benefit from the mechanism of increasing ATCO hours on selected sectors. Figure 12 presents the ACCs with the maximum number of regulations issued due to ATCO staff issues, they represent 80.0% of the number of regulations and 80.8% of the total number of minutes of delay directly generated by staff issue regulations.

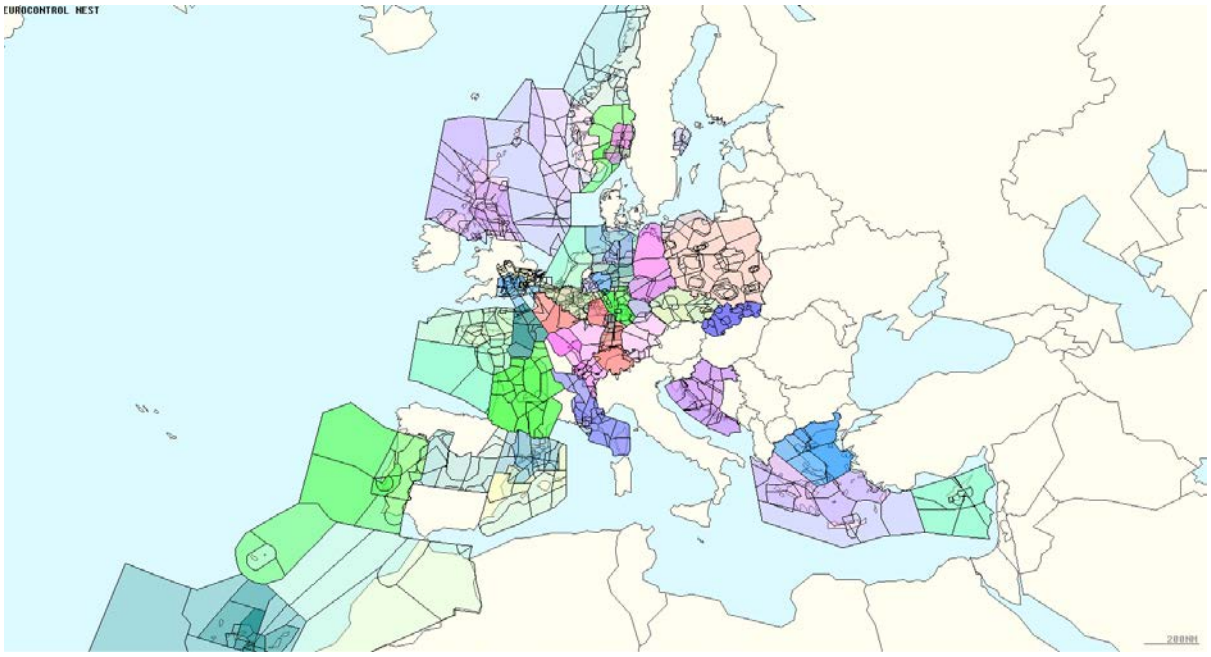


Figure 11. ACCs which reported any ATC Staff ATFM regulation during AIRACs 1313 to 1413. Colours to differentiate ACCs from NEST

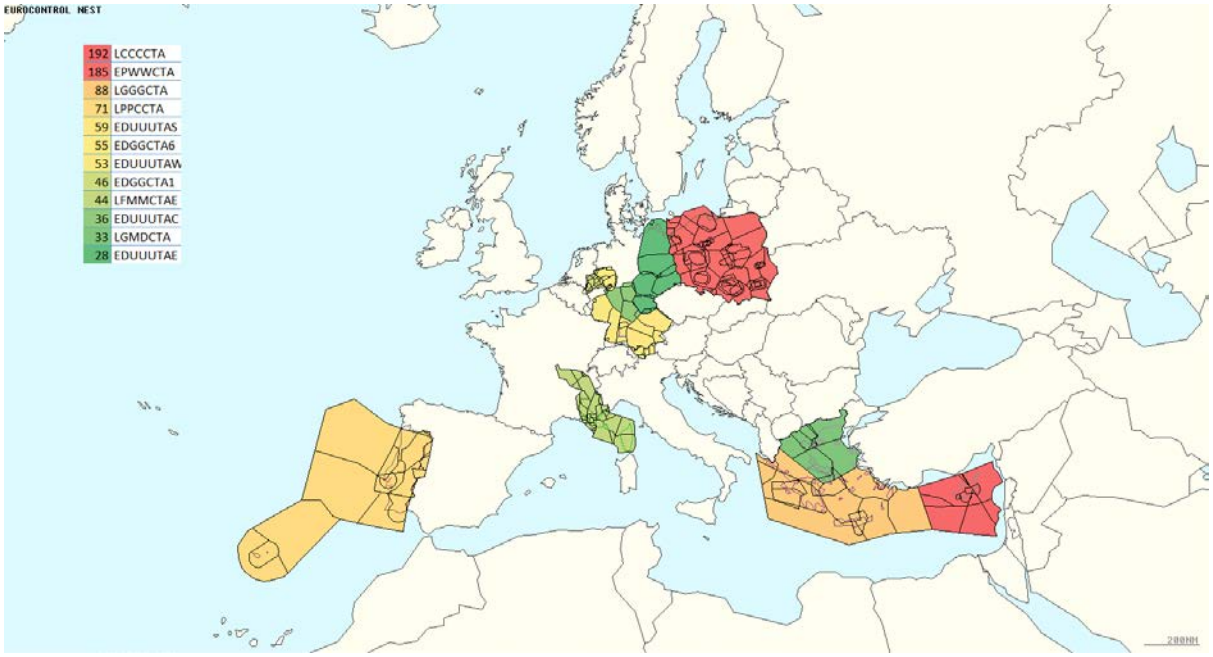


Figure 12. ACCs with the maximum number of ATC staff ATFM regulations

Not all the areas with higher number of regulations and minutes of delay directly attributed to staff issues are of interest in this modelling, but only the ACCS where a different ATCO configuration could lead to benefits. For this reason, the difference between the average ATCOs available during the regulation and the maximum number of ATCOs available in the ACC during the AIRAC of the regulation period has been computed (see Figure 13). This helps us to identify areas that potentially could benefit from a higher number of controllers during the regulations (i.e. during the regulation there are less controllers available than at the maximum at the ACC and therefore, an airspace

configuration with more controllers could be implemented). Figure 15(a) presents the ACCs with a higher gap. Note that some ACCs are not present in the previous analysis of ACCs with maximum number of ATCO staff related regulations (e.g., LECMCTAS has a the second highest average gap of controllers available with respect to the maximum at the ACC (10.0) but only issued 3 regulations during the analysed period).

Comparing the staff available with respect to the maximum available at the ACC allow us to identify ACCs which could have a sectorisation with more ATCOs that might deliver higher capacity during the regulation periods. However, considering the maximum staff available during the whole AIRAC can be too costly, for example, in the regulation presented in Figure 13 the maximum number of staff available was 2 days before the regulation and the regulation is short enough as to not justify the cost of extra staff to solve it. For this reason, an available staff window has been defined as twice the length of the regulation before and after the regulation. This window will allow us to compute the extra staff available within a period of time around the regulations (see Figure 15 and Figure 13). As presented in Figure 14(b), the ACCS which in average have a higher staff available are different than when the whole AIRAC is considered. Note, however, that there are ACCs which have a small number of regulations due to staff shortage (e.g., LEMCTAS).

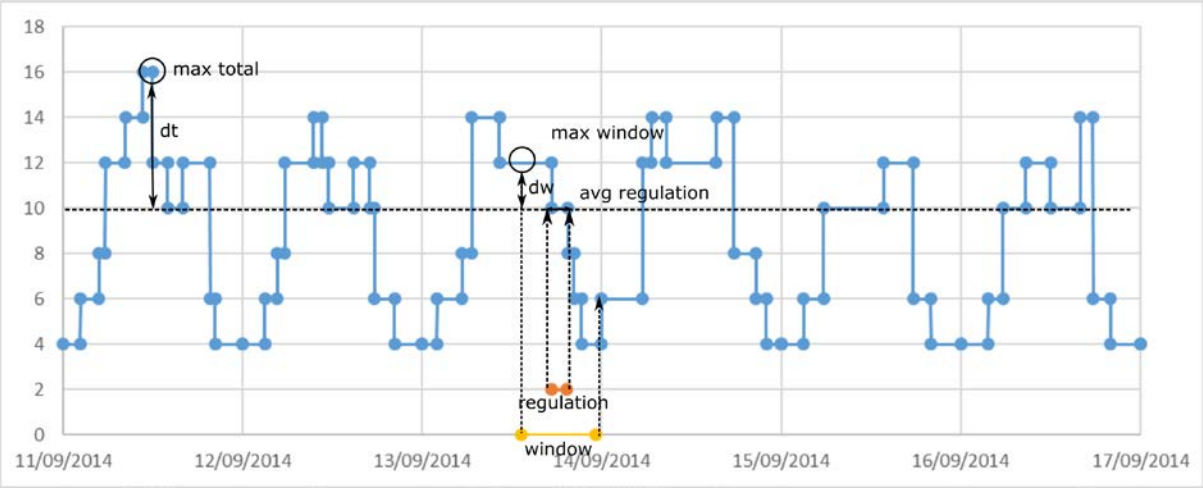
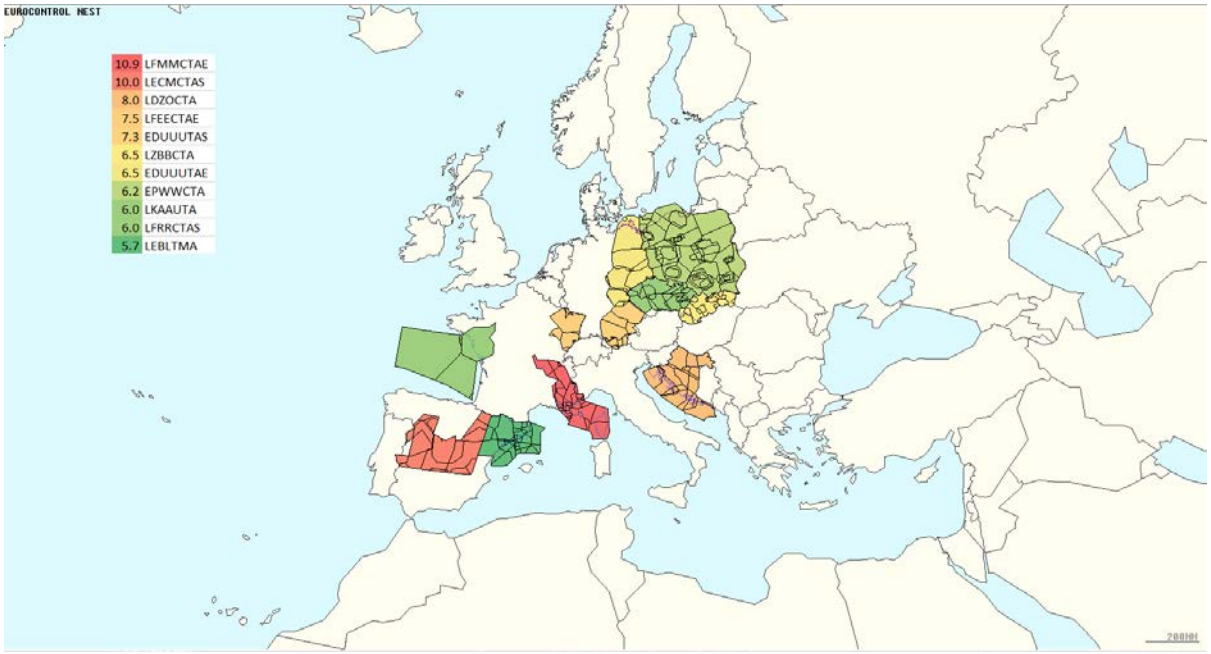
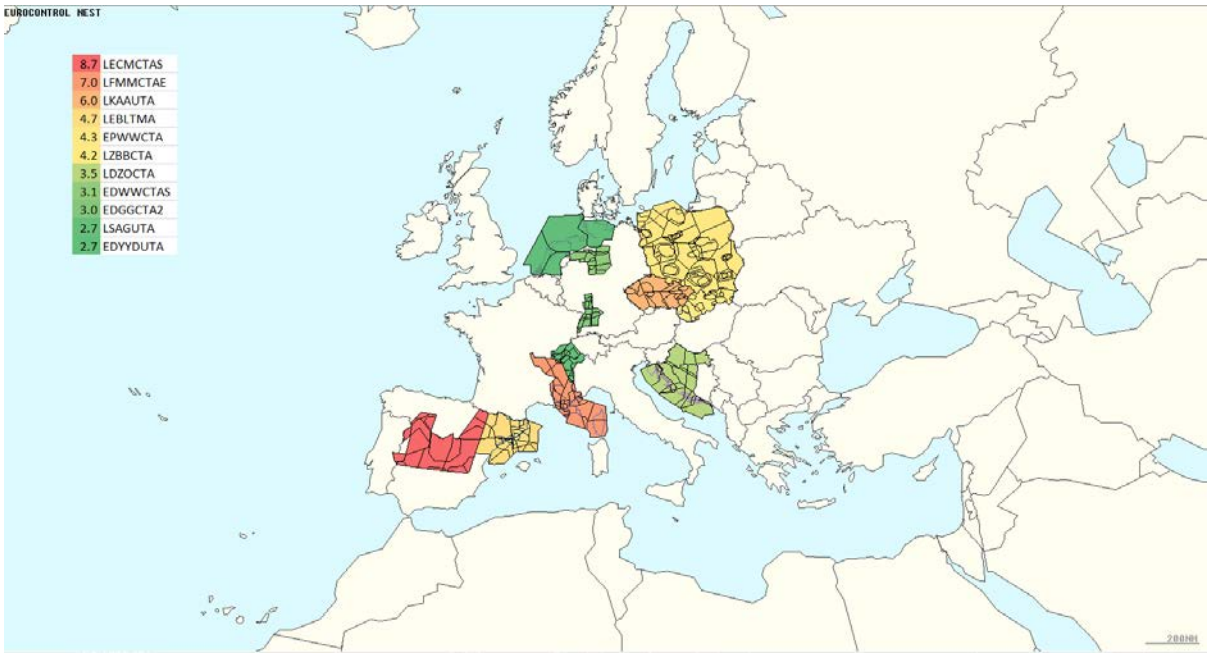


Figure 13. Extra staff available for a given regulation



(a) with respect to the maximum staff available in ACC during the AIRAC



(b) with respect to the maximum staff available within the available staff window

Figure 14. Staff declared available at ACC

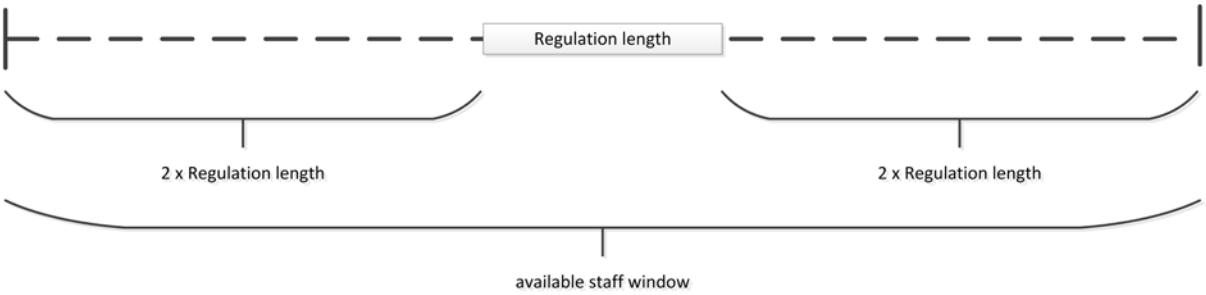


Figure 15. Definition of available staff temporal window around a given regulation

The final approach to identify the ACCs that might benefit the most from the mechanism of extra staff at selected ACCs/sectors is to compute the total delay that has been generated by regulations that where indicated as staff shortage but that in their available temporal window have a gap of four or more ATCOs. Four ATCOs has been selected as it would allow us to open an airspace configuration with two extra sectors that might help to mitigate the problem. These values have been also computed by regulations that are issued due to capacity and/or ATC routing. The idea is that those regulations might also benefit from extra staff in place. Table 6 presents the airspace with the most number of regulations by staff related issues along with the total delay directly generated by those regulations; for each of those regulations, the values of the of average available ATCOs gap between the regulations and the maximum in the AIRAC and in the staff temporal window is also shown; the number of regulations that might benefit from extra staff in the ACC include also the regulations that where declared due to capacity or ATC routing and that have a gap within the window of four air traffic controllers or more; finally, the minutes of delay that could be potentially saved with extra staff due to staff issue regulations and for all the regulations that could benefit from extra staff are shown.

Table 6. Analysis of airspace with maximum number of regulations due to staff shortage[1]

Airspace	Number regulations due to Staff	Total delay generated in airspace due to Staff (min)	Average shortage of available staff during regulation with respect to maximum in Airspace	Average shortage of available staff during regulation with respect to maximum in window	Number of regulations with staff below or equal to 4 short with respect to maximum in window for regulations that might benefit from extra staff	Total delay generated in airspace due to Staff where available staff is lower or equal t 4 with respect to maximum available in window (min)	Total delay generated in airspace due to regulations that might benefit from extra staff where available staff is lower or equal t 4 with respect to maximum available in window (min)
EPWWCTA	185	102,869	6.2	4.3	208	66,966	126,597
LFMMCTAE	44	22,750	10.9	7.0	214	15,166	118,117
LPPCCTA	71	89,987	4.5	2.1	36	41,445	58,157
LCCCCTA	192	144,102	3.0	1.6	76	10,407	40,359
LGGGCTA	88	80,347	4.2	2.1	24	7,954	15,458

[1] Source: own elaboration based on DDR2 data

LGMDCTA	33	24,415	2.9	1.7	30	4,619	9,712
EDUUUTAS	59	23,175	7.3	1.1	66	5,165	6,351
EDUUUTAW	53	18,305	3.5	1.2	8	2,031	2,716
EDGGCTA1	46	11,803	1.1	1.0	8	226	1,407
EDUUUTAC	36	11,665	4.8	1.4	6	982	1,197
EDUUUTAE	28	8,946	6.5	0.8	3	449	449
EDGGCTA6	55	15,249	2.7	1.4	2	417	417

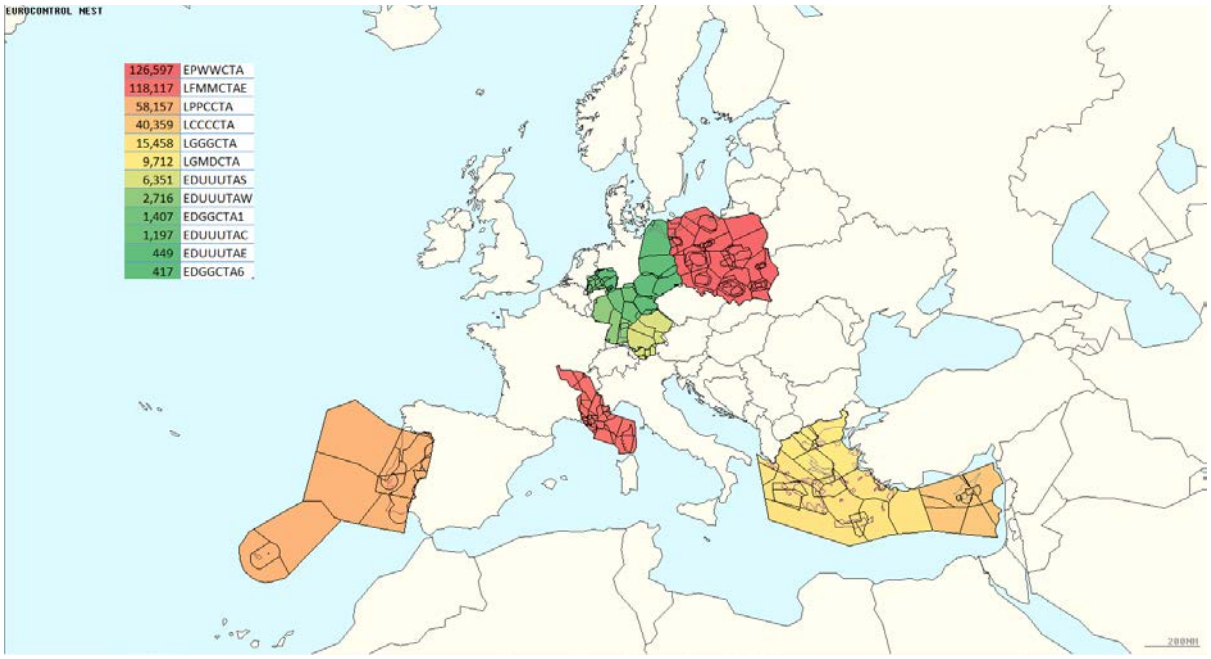


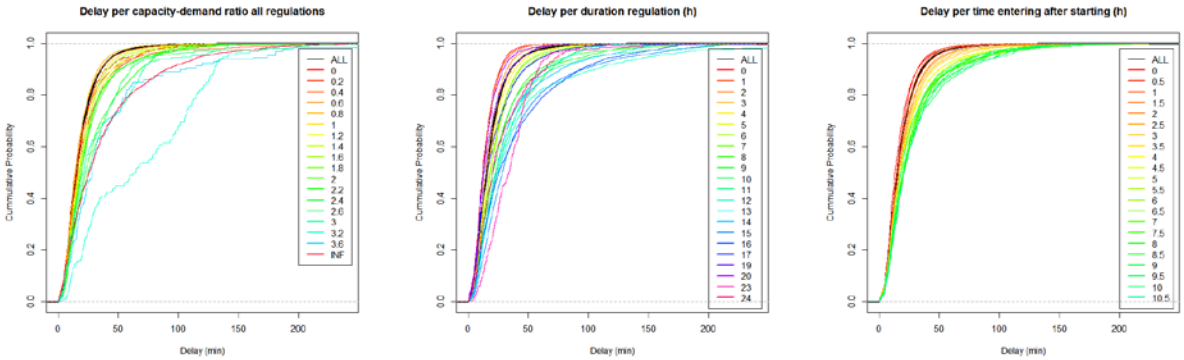
Figure 16. ACCs with the most minutes of delay from regulations of staff, capacity and ATC routing

Figure 16 presents the ACCs with more potential minutes of delay that could be recovered thanks to the implementation of more ATCOs at selected hours during the AIRACs analysed. Finally, it is worth noticing that as the criteria to select a given regulation is that there is a gap of four ATCOs or more within the regulation window, this means that those ACCs might potentially benefit from a different ATCO rostering, if instead ACCs are selected where the gap is small and staff issue regulations are flagged, the ACCs which would benefit from new sectorisations and new staff would be identified, however, those ACCs might be more prone to issue regulations as capacity problem instead of staff.

The previous description will help to select in which ACC and sectors the staff related regulations will be issued. Based on the analysis of the regulations it will be possible to define its duration time, setting in this manner the scope of the disruption. And analysis of the delay generated by the staff related ATFM regulations will be performed to define how the delay will be generated due to these types of disruptions.

Three parameters have been selected to see if there is a relationship between them and the delay generated that could be used to model the regulations:

- the demand/capacity ratio of the airspace where the regulation is implemented, if a regulation has different periods of times with different capacities, the weighted average (based on the duration of each period) has been considered. Higher flights delays when the capacity/demand is higher could be expected.
- the total duration of the regulation. Longer regulations might lead to higher amount of delay for individual flights.
- the time when a flight enters the controlled traffic volume with respect to when the regulation started. The idea is that if a flight enters the regulated airspace later it might have higher probabilities of getting higher delays as some of the slots are already assigned.



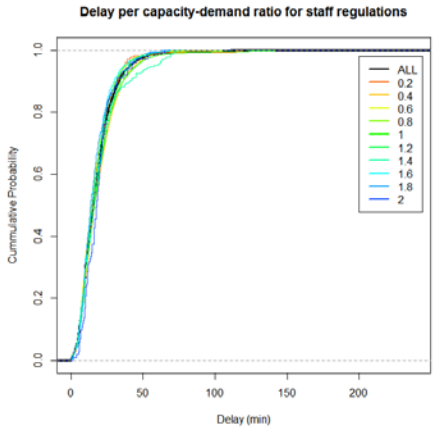
(a) Capacity/demand ratio (b) ATFM regulation duration (c) Time entering the regulated airspace

Figure 17. Cumulative probability distributions for ATFM delay generated, except industrial actions

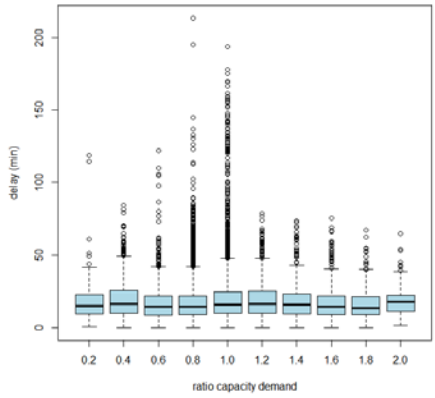
Figure 17 shows the cumulative probability distributions for the ATFM delays generated in the AIRACs periods 1313 to 1413 by all the ATFM regulations except for the issued due to industrial actions. It can be seen that in general higher capacity/demand leads to higher probabilities of higher delays assignments, however, it is not a smooth transition; the duration of the regulation has an impact on the probability of getting assigned higher delays per flight, it is worth noticing that it seems that very long regulations have actually a probability of assigning delays to flights similar to shorter regulations, this might be due to the fact that very long regulations might lead to other measures to avoid the airspace (re-routings or cancellations). Finally, the time when the regulation is entered seems to have a significant role on the probabilities of getting assigned higher delays per flight in a very smooth evolution on the cumulative probabilities distributions functions.

To model the regulations due to staff issues, Figure 18, presents the same results but only for the regulations that were issued due to ATC staff shortage reasons. As presented, for these types of regulations, the ratio between capacity and demand seems not to be as important as the other parameters to define the probability of getting a given amount of ATFM delay.

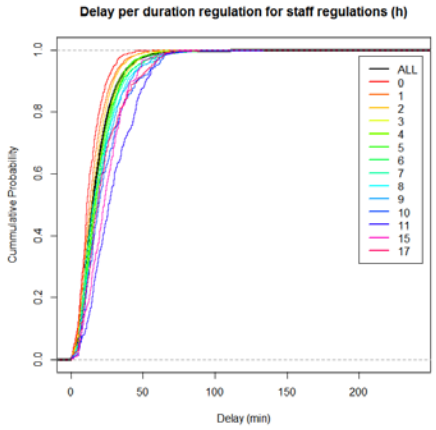
These probabilities of assigning delay will be analysed and modelled to generate realistic disruptions due to ATC staff shortage.



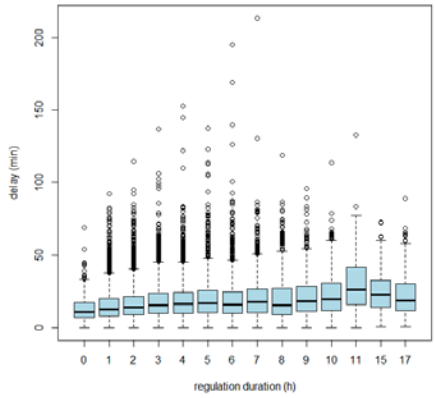
(a) Cumulative probabilities for delay based on capacity/demand ratios



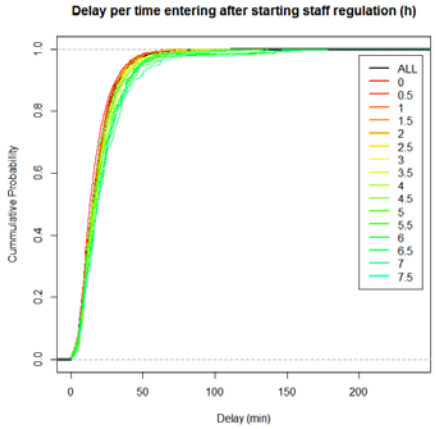
(b) Delay based on capacity/demand ratios



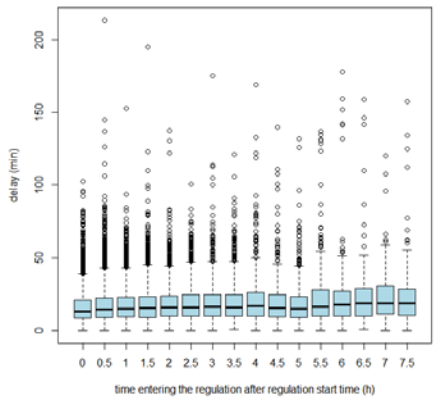
(c) Cumulative probabilities for delay based on regulation duration



(d) Delay based on regulation duration



(e) Cumulative probabilities for delay based on time when entering the regulation



(f) Delay based on regulation duration

Figure 18. Delay generated by ATC staff shortage regulations classified by different parameters

4.3.3 Industrial actions (ATC)

A similar analysis has been done for the industrial actions as for the staff capacity shortage disruptions. As seen in Table 7 (and in Figure 19), during the period from AIRAC 1313 to AIRAC 1413, France aggregated the maximum number of regulations and delay due to industrial actions. It accounts for 99.3% of the total delay directly generated due to industrial actions and for 97.7% of all the regulations issued for that reason.

Table 7. Analysis of airspace with max no. of minutes of delay due to industrial actions regulations

Airspace	Number regulations due to Industrial actions	Total delay generated in airspace due to industrial actions (min (% over total))
LFMMCTAW	54	208,525 (38.3%)
LFRRCTAE	40	72,780 (13.4%)
LFRRCTAS	48	67,936 (12.5%)
LFMMCTAE	41	61,323 (11.3%)
LFRRCTAN	47	51,683 (9.5%)
LFBBCTA	82	30,094 (5.5%)
LFFFCTAW	73	16,228 (3.0%)
LFEECTAE	18	15,739 (2.9%)
LFFFCTAE	37	8,124 (1.5%)
LFEECTAN	19	7,012 (1.3%)

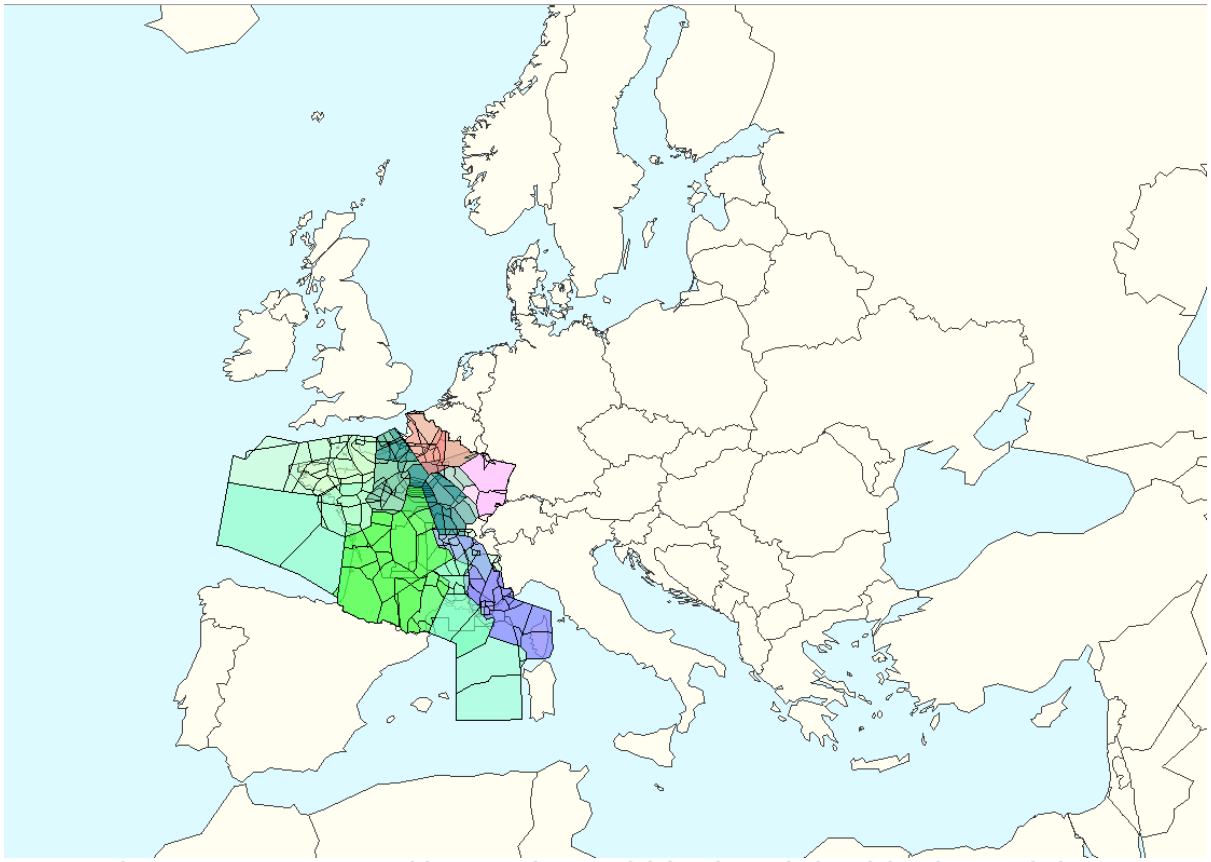
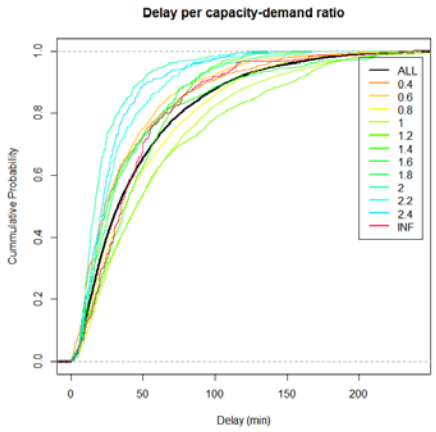
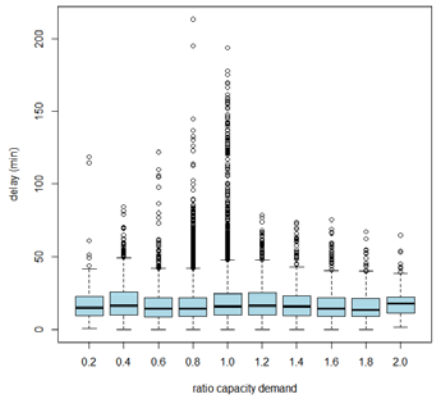


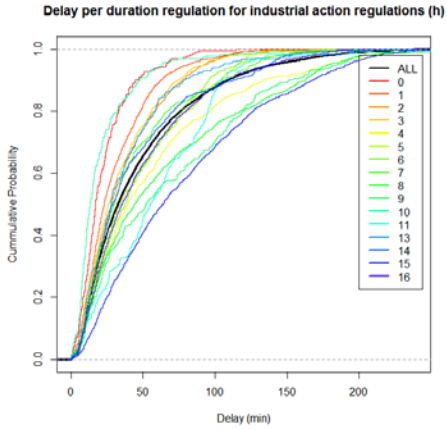
Figure 19. Top ten ACCs with more minutes of delay due to industrial action regulations



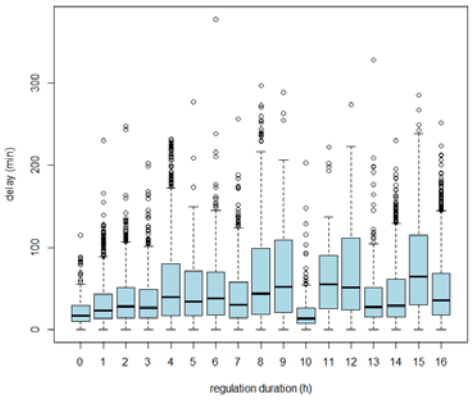
(a) Cumulative probabilities for delay based on capacity/demand ratios



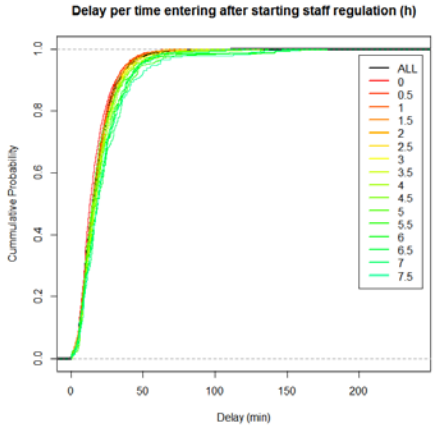
(b) Delay based on capacity/demand ratios



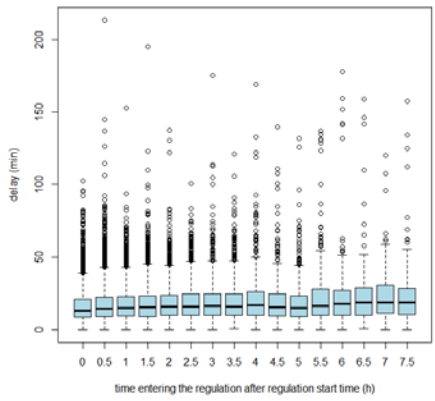
(c) Cumulative probabilities for delay based on regulation duration



(d) Delay based on regulation duration



(e) Cumulative probabilities for delay based on time when entering the regulation



(f) Delay based on regulation duration

Figure 20. Delay generated by industrial actions regulations classified by different parameters

A similar approach as for the ATC shortage disruptions will be used to model the impact of industrial action regulations. Figure 20 shows the cumulative probabilities and the delays for the capacity/demand rations, the regulation duration and the time when the flight enters the regulation for the industrial action regulations. Note that in this case, the capacity/demand ratio seems to have a role on the probability of getting a given amount of delay assigned.

Finally, it is worth mentioning that as presented in Figure 6(a), industrial actions tend to generate higher delay per delayed flight that the other regulations. Its usual predictability and large spatial scope might lead to on one hand a higher use of the airspace (i.e., the capacity declared is closest to the maximum and therefore the delays are higher) and also to more cancellations. Therefore, the analysis of the cancellations that the industrial action regulations generate should be estimated to incorporate them in the model.

4.3.4 Meteorological events with local effects on airports

In this case a particular day with significant disruptions at airports due to weather will be selected as a base to analyse the delay that this type of disruption will generate. Two different delays will be modelled at two different time scopes. First some airports might release ATFM regulations due to a lack of capacity for weather reasons. In that case, a similar methodology as the one described for the regulations due to ATC staff shortage might be used to assign the delay. Secondly, in the presence of weather disruptions high tactical delay at departures and arrivals might be generated. This delay generated on ground and at arrival management will be also need to be modelled.

The intensity (delay generated) and scope are paramount for a properly representation of these type of disruptions. In the case of weather phenomena its temporal evolution is also relevant.

5 Delay cost modelling

5.1 Introduction to the delay cost modelling

In order to be able to assess the mechanisms of Section 4.2 under the disturbances of Section 4.3, it is necessary to model the corresponding cost impacts. These are modelled as costs of delay to the airline, across a range of delay durations, according to 'low', 'base' and 'high' cost scenarios, for the year 2014, by various phases of flight. These values replace previous costs of delay published by the University of Westminster for the reference year 2010.

The main costs of delay to the airline are comprised of passenger, fuel, maintenance, crew and (strategically) fleet costs. This section offers a summary of the *results* to date of the tactical delay cost modelling, which has substantially progressed since Deliverable 2.1 (Theoretical Stochastic Layered Network Model). Although we do not repeat here the methodological details already presented in the latter, these results will also be published as a separate, stand-alone reference publication with a user 'how-to' guide. This will include:

- reporting on, and tabulations of, the strategic costs (not required for the tactical delay cost impacts of ComplexityCosts);
- reporting on, and tabulations of, the statistical reactionary costs (calculated *explicitly* in the ComplexityCosts model);
- any adjustments to the passenger cost of delay to the airline, following the airline consultation on these costs (see Section 5.4).

Although the passenger cost of delay is often a dominating delay cost for operators, there remains limited evidence supporting the calculation of such costs. These costs have therefore gone out to airline consultation, as discussed below. The other costs, e.g. relating to fuel, maintenance and crew, are more readily quantifiable from (published) data sources: there will, therefore, not be a separate consultation process on these elements.

5.2 New aircraft to be included in the model

The cost models comprise 15 aircraft, thus adding three new aircraft to the previously modelled set, increasing the proportion of 2014 flights covered within the CFMU area to almost 63%. The rationale for the selection of the three new aircraft was presented in Deliverable 2.1 (Theoretical Stochastic Layered Network Model). Table 8 (reproduced from D2.1) shows the updated list of aircraft types, with typical seating ranges (excluding outliers) and MTOWs (weighted by flights during 2010-2014).

A minor problem has been encountered with the E190 in some datasets due to its ICAO aircraft type designator covering three aircraft models (Embraer E190, the larger E195 and derived Lineage 1000 business jet). Data cleaning, for example using tail numbers to distinguish between 'E190' aircraft, enables the correct maximum seating per aircraft to be assigned, required by the on-going passenger allocation process.

Sourcing sufficient operating cost data for the three new aircraft has been difficult to achieve despite an in-depth literature search and limited consultation with industry. Maintenance, crew and fleet (the latter not reported here) costs per block hour for DH8D, E190 and A332 aircraft have been estimated by comparison with similar aircraft types. These cost ratios have been furnished from literature over multiple years, including Airline Monitor (ESG Aviation Services) and other industry publications. For example, excluding outliers, the average maintenance block hour cost of the E190 was found to be 10% lower than the B735 between 2010 and 2014.

Table 8. Updated core aircraft types in airline cost of delay models

Aircraft	Aircraft type	Typical seat range[1]	MTOW (tonnes)[2]
B733	B737-300	116-148	60.4
B734	B737-400	134-168	65.4
B735	B737-500	96-132	55.6
B738	B737-800	144-189	74.6
B752	B757-200	160-232	107.8
B763	B767-300ER	192-270	181.5
B744	B747-400	275-436	392.8
A319	A319	118-156	67.1
A320	A320	136-180	74.0
A321	A321	169-220	86.7
AT43	ATR42-300	42-50	16.8
AT72	ATR72-200	62-72	22.2
DH8D	Dash 8 Q400	70-78	29.1
E190	ERJ 190-100	93-106	48.8
A332	A330-200	211-303	230.5

[1] Typical seat range for the global fleet 2010; aircraft with unusual seat configurations excluded.

[2] Weighted average MTOW of flights in the CFMU area 2010-2014 (personal communication, 2015).

5.3 Fuel (and carbon), maintenance and crew costs

5.3.1 Fuel

The cost of fuel is shown in Table 9. The base cost has increased by 0.2 EUR/kg since the earlier reporting in 2010, following global market trends in oil prices (discussed in Deliverable 2.1). Fuel burn is now included in the at-gate calculations, capturing APU usage on the ground, whereby it is assumed that for the base cost scenario the APU is used for 20 minutes during turnaround and for the high cost scenario for 30 minutes. These values are distributed uniformly across the whole turnaround process, e.g. 20 minutes over a 100 minute turnaround means that 20% APU usage, on average, is applied. This equates to the APU running for around 25% of the time under the base cost scenario and 50% of the time for the high cost scenario, as averaged across all the aircraft (of course the percentage used for widebodies is much lower due to the longer turnaround times). No APU fuel burn is associated with the low cost scenario. Fuel burn rates for taxi, en-route and arrival management phases were as reported in Cook and Tanner (2011), with APU fuel burn sourced from the Airport Cooperative Research Program (TRB, 2012) and supplementary data for the new aircraft types sourced primarily from BADA.

Table 9. Cost of fuel

Scenario	Cost of fuel / kg (EUR)
High	0.9
Base	0.8
Low	0.7

5.3.2 Carbon

It was concluded in Deliverable 2.1 (Theoretical Stochastic Layered Network Model), where an exhaustive analysis was presented, that the cost of carbon could be neglected in these calculations, as it effectively comprised a small percentage variation in the cost of fuel, much smaller than actual variations in the fuel price itself.

5.3.3 Maintenance

Maintenance costs are incurred by aircraft whether on the ground or en-route, and need to be proportionally allocated across flight phases: at-gate, taxi and airborne (en-route and arrival management) according to workload. For example, at-gate maintenance costs are low compared with those accrued during the more intensive take-off phase.

The reassessment of the 2010 reference values (Cook and Tanner, 2011) and their initial update to 2014 costs was described in Deliverable 1.2. This previous review of the trend in maintenance block-hour costs reported to ICAO (ICAO, 2014), combined with a selection of European airline financial returns suggested (i) no change between 2010 and 2011, followed by (ii) a 15% increase on maintenance block-hour costs across all scenarios to adjust the reference values to 2014 values. Although slightly more recent financial data have been published since this review (ICAO, 2015), summary maintenance costs covering 2014 are predominantly only available via individual airlines' financial returns. Ten European airlines' annual financial returns[2] covering 2011-2014 have been examined (in addition to ICAO, 2015) allowing the previously reported increase to be refined across the three cost scenarios. Although further refinements may be necessary, the new adopted changes to maintenance block-hour costs per scenario are (2011-2014):

- **Base: 15% increase**, based on the overall average maintenance cost increase across all ten airlines (14%);
- **Low: 5% increase**, although two airlines reported no change and -1% maintenance cost decrease 2011-2014, a small increase has been adopted;
- **High: 25% increase**, derived from the highest maintenance cost increase of two airlines.

With maintenance block-hour costs updated to 2014-Euro values, the tactical (and strategic) costs are derived. The fixed costs, such as the overhead burden, are removed leaving the time-based costs to be apportioned across flight phases (see Table 10 to Table 12). The updated tactical costs are 13-14% higher than previous reference values (base scenario across the at-gate, taxi and airborne phases). With only a small fluctuation due to other updated inputs (e.g. 2014 rotations per day and service hours), the main driver of these changes is the increase in maintenance block-hour costs.

[2] Other airline returns were excluded if missing any of their 2011, 2012, 2013 or 2014 costs; airline returns were also excluded if significant reporting changes or a significant increase/decrease to fleet composition occurred during this timeframe (thus affecting maintenance provision).

Table 10. At-gate: tactical maintenance costs (per minute)

Aircraft	Low scenario	Base scenario	High scenario
B733	0.2	0.5	0.7
B734	0.2	0.5	0.7
B735	0.2	0.5	0.6
B738	0.2	0.5	0.7
B752	0.3	0.6	0.8
B763	0.4	0.8	1.3
B744	0.8	1.2	1.4
A319	0.2	0.6	0.8
A320	0.2	0.5	0.8
A321	0.3	0.6	0.8
AT43	0.1	0.2	0.3
AT72	0.1	0.3	0.4
DH8D	0.1	0.3	0.4
E190	0.2	0.4	0.6
A332	0.4	0.9	1.4

All costs are EUR per minute (2014) and exclude overheads.

Table 11. Taxi: tactical maintenance costs (per minute)

Aircraft	Low scenario	Base scenario	High scenario
B733	1.3	3.0	4.1
B734	1.5	3.3	4.4
B735	1.3	2.8	3.8
B738	1.1	2.6	4.2
B752	1.6	3.6	4.8
B763	2.4	4.9	7.7
B744	4.8	6.7	8.1
A319	1.4	3.3	4.5
A320	1.5	3.1	4.9
A321	1.7	3.6	4.9
AT43	0.7	1.5	2.0
AT72	0.9	1.9	2.6
DH8D	0.8	1.8	2.4
E190	1.2	2.6	3.5
A332	2.6	5.2	8.3

All costs are EUR per minute (2014) and exclude overheads.

Table 12. En-route: tactical maintenance costs (per minute)

Aircraft	Low scenario	Base scenario	High scenario
B733	1.6	3.9	5.4
B734	1.9	4.2	5.7
B735	1.7	3.6	4.9
B738	1.5	3.4	5.4
B752	2.1	4.6	6.1
B763	3.1	6.2	9.9
B744	6.2	8.7	10.6
A319	1.8	4.3	6.0
A320	1.8	4.0	6.2
A321	2.2	4.7	6.4
AT43	0.8	1.7	2.3
AT72	1.1	2.3	3.1
DH8D	1.0	2.2	3.0
E190	1.5	3.3	4.5
A332	3.2	6.6	10.4

All costs are EUR per minute (2014) and exclude overheads.

5.3.4 Crew

Flight crew salaries increase by size of aircraft whereas cabin crew salaries are more consistent across all aircraft types. Total cabin crew numbers are driven by the maximum number of seats available, irrespective of the passenger load factor.

In Europe, flight and cabin crew are typically paid through a combination of fixed salaries, supplemented by (relatively) small flying-time payments and cycles-based allowances. An in-house developed crew model (Cook and Tanner, 2011) uses a range of typical crew salaries (low, base and high) to determine salary on-costs (e.g. pension and social security costs) and overtime rates. Crew payment items are separated by the model in order to derive tactical (time-based) and strategic (per service hour) costs.

Crew cost changes since 2010 have been reviewed, with costs/differences sourced from pay agreements, airline financial reporting and industry literature. Although there are examples of pay cuts and pay freezes, modest pay rises (typically 5%) are seen between 2010 and 2014. The previously provisional changes to the range of typical pilot and cabin crew salaries (model input data) reported in Deliverable 1.2 can now be updated as follows:

- **Base: 5% increase**, based on the overall average crew cost increase across a selection of European airlines;
- **Low: no change**, reflecting crew pay freezes rather than pay cuts;
- **High: 10% increase**, although a small number of airlines were reported to have increased pilot salaries by approximately 20% over this timeframe, a 10% increase is more plausible with these outliers excluded.

Table 13 shows the new tactical crew costs that apply to ground and airborne phases. These represent the cost of crewing for additional minutes over and above those planned at the strategic phase. In Europe it is possible for marginal crew costs to be zero, resulting in no additional costs to

the airline. For example, an airborne delay will have no effect on the cost of crew paid by sectors flown as this payment mechanism is cycles-based (note that a large proportion of the crew's pay would be fixed as basic salary).

Maximum service hours for crew are considered, i.e. 14 hours per day for all narrowbody aircraft with higher, per-aircraft type service hours for widebodies. The high cost scenario for the three widebody aircraft includes the cost of spare flight crew, however crew costs do not vary between hub / non-hub airports.

Driven by the increased crew salary ranges, the tactical base and high cost scenarios are 5% and 10% higher than the previous reference values, except for three aircraft. The B734 and B752 have tactical base and high cost scenarios that are 11% and 16% higher than before, whereas the base costs of the B763 are only 1% higher. These are explained by the updated aircraft seating allocations, resulting in the B734 and B752 requiring an extra member of cabin crew, and the B763 requiring one fewer. Total cabin crew numbers are driven by the maximum number of seats available, irrespective of the passenger load factor.

Table 13. Marginal crew costs, ground or airborne (per minute)

Aircraft	Low scenario	Base scenario	High scenario
B733	0	8.9	19.5
B734	0	9.2	20.6
B735	0	8.4	19.0
B738	0	9.5	21.5
B752	0	9.9	20.9
B763	0	13.0	38.0
B744	0	17.5	49.5
A319	0	7.7	16.7
A320	0	8.2	17.7
A321	0	8.2	17.7
AT43	0	5.9	12.7
AT72	0	6.4	14.3
DH8D	0	6.4	14.2
E190	0	7.1	16.6
A332	0	13.8	30.3

All costs are EUR (2014) per minute and include on-costs.

5.4 Passenger cost allocations and airline review process

As mentioned above, although the passenger cost of delay is often a dominating delay cost for operators, there remains incomplete quantitative evidence supporting the calculation thereof. The published literature on such costs and factors likely to influence them (indirectly) have been examined, including a European Commission Impact Assessment published in 2013, focusing on Regulation 261, although the extent to which quantitative inputs can be used from other reporting methods to update the values previously adopted by EUROCONTROL is very limited. Ultimately, for both the passenger hard and soft costs of delay to the airline, simple inflationary increases have been applied to the 2010 costs, in parallel to updated seat, load factor and passenger allocations for the 15 aircraft types considered. Table 14 presents the total costs of passenger delay to the airlines, by delay duration and aircraft type, for the base cost scenario, in 2014-Euros (the costs are the same for all phases of flight, so only one table is needed). Compared with the previously reported values for 2010, the **average increase is 20%**. Most of this increase has been driven by increasing passenger densities on European flights.

In view of both the more limited evidence for these estimates, and their domination of the total cost of delay, these values have gone out to an airline consultation process, running from 18 August 2015 – 02 October 2015. (This 38 page document is available on request from the ComplexityCosts team. A detailed derivation is given, with differentiation by hard and soft costs, and by cost scenario.) Over 400 airlines and airspace users have been contacted, with a strong European focus. Evidence-based feedback will be taken into consideration regarding any required revisions to these costs, before they are deployed in the ComplexityCosts model.

Table 14. Total cost of passenger delay by delay duration and aircraft type (base cost scenario)

Delay (mins)	5	15	30	60	90	120	180	240	300
B733	40	250	910	3 320	6 800	11 110	22 180	36 370	53 520
B734	40	290	1 040	3 780	7 750	12 670	25 280	41 470	61 020
B735	30	230	810	2 950	6 040	9 860	19 690	32 290	47 510
B738	40	330	1 170	4 250	8 700	14 220	28 390	46 570	68 520
B752	50	400	1 420	5 180	10 610	17 340	34 610	56 770	83 520
B763	70	490	1 760	6 420	13 150	21 490	42 900	70 360	103 530
B744	110	790	2 850	10 360	21 220	34 670	69 220	113 530	167 050
A319	40	270	960	3 510	7 180	11 730	23 420	38 410	56 520
A320	40	310	1 110	4 030	8 260	13 500	26 940	44 190	65 020
A321	50	380	1 350	4 900	10 040	16 400	32 750	53 710	79 020
AT43	10	90	310	1 120	2 290	3 740	7 460	12 240	18 010
AT72	20	120	440	1 610	3 300	5 400	10 780	17 680	26 010
DH8D	20	140	490	1 770	3 620	5 920	11 810	19 380	28 510
E190	30	180	660	2 390	4 890	7 990	15 960	26 170	38 510
A332	80	550	1 980	7 200	14 740	24 080	48 080	78 860	116 030

5.5 Provisional, primary delay cost values for 2014

The provisional, total costs of primary delay by phase of flight for the base cost scenarios are shown in the following four tables. In ComplexityCosts, it is likely that the at-gate (ATFM and turnaround delay effects) and en-route (dynamic cost indexing effects) costs will play the most prominent role. Compared with the 2010 values, the 2014 at-gate costs have increased by **18%**, simply averaged across all the cells (excluding the new aircraft of the lower three rows). This has been driven mainly by the increase in passenger delay costs, summarised in Section 5.4. The increases for the B763 and B744 are lower, with the former values only slightly increased relative to 2010. (This is driven by slight *falls* in passenger densities for the B763 over this period, and relatively lower increases for the B744 densities, thus dampening the increase in passenger costs observed for the other (original) aircraft types.) The increase for the en-route costs similarly averages at **22%**, this time with an additional contribution from the increasing fuel price (see Section 5.3.1), again with less marked increases for the same widebodies.

Table 15. At-gate: total cost of delay by delay duration and aircraft type (base cost scenario)

Delay (mins)	5	15	30	60	90	120	180	240	300
B733	80	400	1 200	3 900	7 670	12 280	23 930	38 710	56 440
B734	90	440	1 340	4 390	8 650	13 870	27 090	43 870	64 020
B735	80	360	1 090	3 500	6 870	10 970	21 350	34 510	50 280
B738	100	490	1 490	4 900	9 680	15 520	30 340	49 160	71 760
B752	110	560	1 740	5 820	11 570	18 620	36 540	59 340	86 740
B763	140	710	2 190	7 280	14 430	23 200	45 460	73 780	107 800
B744	200	1 080	3 420	11 500	22 930	36 960	72 650	118 100	172 760
A319	80	400	1 230	4 040	7 980	12 800	25 030	40 550	59 200
A320	90	450	1 390	4 590	9 100	14 620	28 620	46 430	67 820
A321	100	520	1 630	5 470	10 890	17 530	34 440	55 970	81 850
AT43	40	180	490	1 490	2 850	4 490	8 580	13 730	19 880
AT72	50	230	650	2 030	3 920	6 230	12 020	19 330	28 080
DH8D	50	240	700	2 190	4 250	6 760	13 080	21 060	30 620
E190	70	300	900	2 870	5 620	8 960	17 410	28 110	40 930
A332	150	780	2 430	8 110	16 100	25 900	50 810	82 500	120 580

Table 16. Taxi: total cost of delay by delay duration and aircraft type (base cost scenario)

Delay (mins)	5	15	30	60	90	120	180	240	300
B733	150	600	1 600	4 690	8 860	13 850	26 290	41 860	60 370
B734	160	660	1 770	5 240	9 940	15 580	29 660	47 310	68 320
B735	150	570	1 500	4 340	8 120	12 640	23 860	37 850	54 460
B738	160	670	1 860	5 630	10 770	16 980	32 530	52 080	75 410
B752	210	860	2 350	7 030	13 380	21 030	40 150	64 160	92 760
B763	260	1 070	2 910	8 720	16 600	26 090	49 800	79 560	115 020
B744	420	1 730	4 720	14 110	26 840	42 170	80 470	128 530	185 790
A319	140	570	1 570	4 720	9 000	14 160	27 070	43 270	62 600
A320	160	660	1 820	5 450	10 390	16 330	31 200	49 870	72 110
A321	170	720	2 030	6 270	12 090	19 140	36 860	59 190	85 880
AT43	70	260	650	1 810	3 320	5 120	9 530	15 000	21 460
AT72	80	320	840	2 410	4 500	6 990	13 160	20 850	29 980
DH8D	80	330	880	2 560	4 800	7 490	14 180	22 530	32 450
E190	120	460	1 200	3 480	6 520	10 170	19 220	30 530	43 950
A332	310	1 240	3 350	9 950	18 870	29 590	56 340	89 880	129 800

Table 17. En-route: total cost of delay by delay duration and aircraft type (base cost scenario)

Delay (mins)	5	15	30	60	90	120	180	240	300
B733	260	930	2 270	6 040	10 880	16 550	30 330	47 250	67 110
B734	270	970	2 400	6 520	11 850	18 130	33 480	52 390	74 670
B735	240	850	2 060	5 450	9 790	14 870	27 200	42 300	60 030
B738	280	1 030	2 580	7 080	12 950	19 890	36 890	57 900	82 680
B752	350	1 280	3 180	8 700	15 890	24 380	45 170	70 840	101 120
B763	480	1 730	4 230	11 360	20 550	31 360	57 700	90 100	128 200
B744	890	3 150	7 560	19 780	35 350	53 520	97 490	151 220	214 160
A319	250	910	2 250	6 070	11 030	16 860	31 120	48 670	69 340
A320	260	960	2 420	6 650	12 180	18 730	34 790	54 650	78 100
A321	300	1 130	2 850	7 910	14 550	22 420	41 770	65 740	94 070
AT43	80	290	710	1 920	3 490	5 340	9 870	15 450	22 020
AT72	100	380	960	2 640	4 850	7 450	13 860	21 790	31 150
DH8D	120	440	1 090	2 980	5 440	8 340	15 440	24 210	34 550
E190	190	680	1 660	4 400	7 900	12 010	21 980	34 200	48 540
A332	540	1 930	4 740	12 730	23 040	35 150	64 670	100 980	143 690

Table 18. Arrival mgmt: total cost of delay by delay duration and aircraft type (base cost scenario)

Delay (mins)	5	15	30	60	90	120	180	240	300
B733	220	820	2 050	5 590	10 200	15 640	28 980	45 440	64 850
B734	260	940	2 330	6 370	11 630	17 840	33 050	51 820	73 950
B735	190	710	1 780	4 900	8 960	13 760	25 540	40 100	57 270
B738	260	970	2 460	6 830	12 570	19 380	36 130	56 880	81 410
B752	300	1 120	2 870	8 080	14 960	23 140	43 310	68 370	98 020
B763	450	1 640	4 060	11 010	20 030	30 670	56 670	88 720	126 470
B744	670	2 470	6 200	17 070	31 280	48 090	89 340	140 350	200 580
A319	230	850	2 140	5 850	10 700	16 420	30 450	47 790	68 240
A320	260	950	2 390	6 610	12 120	18 640	34 660	54 480	77 880
A321	280	1 070	2 730	7 670	14 190	21 940	41 050	64 780	92 870
AT43	80	290	710	1 910	3 480	5 330	9 860	15 430	22 000
AT72	100	360	920	2 560	4 720	7 290	13 620	21 460	30 740
DH8D	120	440	1 090	2 980	5 440	8 340	15 440	24 210	34 550
E190	190	670	1 630	4 330	7 800	11 870	21 780	33 930	48 210
A332	440	1 630	4 130	11 510	21 210	32 720	61 030	96 120	137 610

6 Verification and Credibility assessment

6.1 Verification and validation in context

Where ***validation*** procedures ensure we are building the right model for the questions that ComplexityCosts poses, ***verification*** ensures that the system is built correctly, without errors or software bugs (EUROCONTROL, 2010). Different models are developed to different maturity levels: basic principles of phenomena, application formulated, development of a proof of concept, etc., as formalised in Table 19. The labels in the second column are those cited by Gilbert (2008) in the context of agent-based modelling, although still useful in a more descriptive general context.

Table 19. Model levels

Level of model	Nomenclature	Key characteristics
Level 1	Abstract	Demonstrates a basic process
Level 2	Middle range	Characterises a particular phenomenon such that conclusions can be applied more widely
Level 3	Facsimile	Reproduces a particular phenomenon as exactly as possible, usually with the intention of prediction or retrodiction of another state, based on given changes

Different stages of concept development require different verification and validation techniques. According to this type of framework, ComplexityCosts is somewhere between Level 2 and Level 3, with obvious practical limitations in trying to reproduce the full, European air transport within such a timescale and budget. The project aims to reach TRL2, defined by NASA (2010) as "technology concept and/or application formulated" (NASA developed the original TRL definitions) and echoed by SESAR (2015) in its Exploratory Research formulation, whereby:

- a theory and scientific principles are applied to a very specific application area to perform the analysis and to define the concept of costs trade-offs;
- the characteristics of investment mechanisms and disturbance types are described, evaluating the potential cost benefits of the former;
- analytical tools are developed for simulation and analysis of the mechanisms under disturbance;
- uncertainty is taken into account, and integrated into the model.

Both formal **validation and verification start at TRL5**. However, some elements of validation and verification are recommended at any TRL. This section thus elaborates on different actions that have been carried out during the project, to ensure validation and verification are executed at different stages to avoid potential major pitfalls. This early stage of the validation, performed at TRL2, should be understood as a **credibility assessment**.

6.2 Credibility assessment

The credibility assessment shall address the basic questions regarding the methodology of the modelling chosen for ComplexityCosts. In particular, a number of questions are addressed:

- Are we building the right model to ***understand the ATM network performance trade-offs for different stakeholder investment mechanisms in the context of uncertainty, under disturbance?***

In order to conform with the previous requirement the model is able to produce a set of stochastic metrics. In fact, most of the metrics are unique in the sense that they are not currently measured by any known system. As was shown in the SESAR WP-E POEM project (Cook *et al*., 2013), some operational changes can only be detected by using passenger-centric metrics. In addition, studying global optima in a complex environment is not logical given the compromised values and trade-offs.

Such metrics can also be used to compare the model outputs with **empirical data**: for example, does the model baseline (without new investment mechanisms in place) produce known delay characteristics, such as departure delay distributions and primary/reactionary delay ratios, of the actual baseline day (wherein no marked disturbances were observed - hence its selection)? Furthermore, a **qualitative sensitivity analysis** can be established, examining relationships between the key metrics. It is quite unlikely that any given mechanism will produce a simultaneous improvement across all metrics, such that if airline delay costs decrease we may expect negative impacts on other factors (such as reactionary delay), which may relate to previously reported trade-offs in the literature.

- Is the **stochastic, event-driven layered network model the right approach**?

As many socio-technological systems, the air transportation system has proven to perform in a non-deterministic way. Human decisions are often unpredictable, system failures are of course of random nature and furthermore many decisions are based on weather forecast which always has some degree of uncertainty associated. Therefore, any model of the air transportation system should tackle uncertainty, or in other words be of stochastic nature.

As any point-to-point transportation system there is a direct translation between the real system and a network model, which makes network theory very powerful in this context. In addition, the hierarchy of the AT system is better represented using a layered network model. Finally, the AT system is a very procedural system, with many sub-processes carried out to produce the overall behaviour. Event-driven paradigm is the perfect match for this kind of system, which adds flexibility to explore new concepts, mechanisms and disruption effects.

- As part of the requirements of such a model, are we including **interacting elements and feedback loops**?

As a system of systems interacting elements are key when modelling the air transportation system. Most of this interaction is in fact one-to-one, but the effects propagate through the entire system, and in many cases, generate feedback loops. In the ComplexityCosts tool, interactions between elements have been carefully designed using class methods and restrictive language and information available to ensure that the simulation is according to the actual system. The event structure and the event manager have been developed so that the ComplexityCosts tool is able to manage continuous feedback and iteration. The software was built using a bottom-up approach, in which each component is individually designed and approved, and then as compiled and joined, the system emerges as a global behaviour.

- Is there an **external** credibility assessment in place?

While this is not a replacement for a validation exercise *per se*, external feedback does strengthen the credibility of the approach. As reported in our Progress Reports: (i) key deliverables have been, and will continue to be, reviewed by members of the SJU airspace users' group; (ii) a dialogue is open with the European Commission with regards to the Pilot Common Project (PCP) cost-benefit analysis, regarding cost inputs; and (iii) a dialogue is open with EUROCONTROL regarding the cost-benefit analysis methodology. These dialogues are in addition to the continuous constructive inputs received from the EUROCONTROL Project Officer, and from the SJU at the Gate Meeting. Furthermore, data integrity and industry consultation is outlined in Section 6.4. Lastly, peer review of the project

methodology through journal and conference publications (see also Section 8) is also of value. Collectively these actions strengthen the entire model's credibility.

6.3 Verification of the model implementation

The verification of the model provide proof that the model has been implemented correctly, according to the specifications previously established by the project. It is not part of the verification to evaluate the conformity of the theoretical models to be implemented which are taken for granted, but rather to evaluate how closely the theoretical models have been implemented. Absolute zero bug state in software programming is just a delusion: errors will always appear and unexpected behaviour is always possible. However, it is the task of verification to ensure that the number of errors is dropped to a minimum and that the unexpected behaviours have the least possible impact on the overall performance.

There are two main verification approaches used in the ComplexityCosts development: dynamic and static testing. Dynamic testing implies running the model and either run a unit test (isolated components or class method) or integration tests (groups of classes and interacting methods).

Dynamic test are usually divided into three categories:

- Functional tests use simplified inputs with known outputs to evaluate critical functionalities of the implementation.
- Structural test use simplified inputs to test the code structure. That is to say inputs may not correspond to a realistic scenario but rather a worst-case scenario. For instance a case of extreme values testing.
- Sequential or random test is an exhaustive test tool in which inputs are thrown into the tested components and outputs are analysed checking for inconsistencies. For instance using a simplified input set with minimal variation to perform a sensitivity analysis (i.e. expected solutions should be continuous and smooth).

On the other hand, static testing does not involve running the model or any part thereof. Rather, it involves source code analysis (coding practices, patterns use ratio, code design, compliance with standardised best practices, etc.) as well as right types for input and output data, parameter matching between methods. Exhaustive static tests have been performed against the ComplexityCosts model, and in some cases have led to improvements and debugging. However the static testing phase is an iterative process that continues during the whole project execution.

6.4 Integrity of model input data

Integrity of the model data concerns both validation and verification. From a validation point of view, the right datasets have to be taken into account to ensure that there are not missing pieces in the modelling tasks. From the verification perspective, the data model has to be implemented following the requirements of the project.

6.4.1 Traffic and passenger model

Data integrity, maintaining the accuracy and consistency of data within ComplexityCosts, is a key component within WP1. All datasets intended for use in the models are first checked, cleaned where necessary, enhanced with data from other sources and tested during the data preparation process.

The following table briefly updates previous reporting by summarising the various on-going data preparation tasks.

Table 20. Data preparation tasks

Dataset	Main source	Preparation tasks
Traffic data	DDR2 (EUROCONTROL); in-house databases*	**Consistency checks**, e.g. tail number tracking throughout the day; **Cleaning**, e.g. identifying and recoding errors; **Exclusions**, e.g. identifying non-passenger IFR flights such as cargo/military/business aircraft; **Enhancements**, e.g. assigning additional required information to flights/aircraft in scope: AO type (i.e. full-service, LCC, charter or regional); maximum number of seats; schedule times; assigned AO (i.e. determining marketing carrier from reported callsign/company/operating company).
Passenger itineraries	In-house databases*; GDS sample; Eurostat	**Consistency checks**, e.g. aligning 2010 itineraries with 2014 flights; **Cleaning**, e.g. assigning itineraries to 2014 flights using partner/alliance airlines; **Exclusions**, e.g. identifying 2010 itineraries that cannot be achieved with 2014 flights; **Enhancements**, e.g. new 2014 itineraries from GDS sample data and Eurostat O/D data.
Delay costs	In-house databases*	**Consistency checks**, e.g. cost changes since 2010 and cost of passenger delay to European airline operations consultation; **Cleaning**, e.g. determining raw maintenance/crew/fleet/passenger costs for 2014; **Exclusions**, N/A; **Enhancements**, e.g. inclusion of additional aircraft within the delay cost models.
Airport list	ACI EUROPE; in-house databases*	**Consistency checks**, e.g. changes since 2010 (i.e. airport closures); **Cleaning**, e.g. determining changes within previously selected ECAC/non-ECAC airports (such as Hamad International Airport replacing Doha International Airport in April 2014); **Exclusions**, N/A; **Enhancements**, e.g. time-zones; IATA coding; geographical coordinates (for GCD calculations).

* In-house databases are maintained from various public and private data sources.

6.4.2 Delay cost models

In Section 5.3, updates on the fuel, maintenance and crew costs are summarised. As also detailed in previous deliverables (*viz*. D1.2 and D2.1), these costs draw primarily on Airline Business (fuel), airline financial reporting and pay agreements (fuel, maintenance, crew), International Civil Aviation Organization datasets (direct maintenance costs and utilisation data for processing the maintenance and crew costs) and EUROCONTROL's Performance Review Unit (operational statistics, including

rotations and flight duration). As discussed in Section 7.4, the passenger cost of delay to the airline draws on multiple data sources. Although a wider, systematic review has been recommended by the University of Westminster for many years, the funding for this is unlikely to materialise. Since these costs of delay often dominate the total cost, this aspect of the delay cost calculations has gone out to an airline consultation process, running from 18 August 2015 – 02 October 2015. Over 400 airlines and airspace users have been contacted, with a strong European focus. Evidence-based feedback will be taken into consideration regarding any required revisions to these costs, before they are deployed in the ComplexityCosts model.

6.4.3 Mechanism costings

These were discussed in Section 4.2. Also, a comprehensive literature review was carried out in Deliverables D1.2 and D2.1. One of the criteria used to select a mechanism to be implemented in ComplexityCosts was the availability of reliable sources for the modelling of the strategic and tactical costs of the mechanism. Drawing on ATM cost effectiveness reports by EUROCONTROL and on ATCO training costs (e.g., EUROCONTROL, 2015a; NATS, 2015; SENASA, 2015), reliable ATCO costs can be estimated; A-CDM is widely implemented throughout Europe and cost-benefits analysis, with a breakdown of the cost per stakeholder, have been undertaken in several airports (e.g., EUROCONTROL, 2008; Deutsche Flugsicherung, 2013). Costs of passenger disruption management software and dynamic cost indexing applications are harder to estimate and, in light of the difficulties of obtaining such costs, we have contacted the major suppliers to the European market. The project team has made assurances to the suppliers regarding disclosure, such that these suppliers are not identified in this report, pending related permissions, in order to maximise the extent to which the suppliers will disclose robust data.

6.4.4 Disturbance modelling

When disturbances are present delay is generated and ATFM regulations might be put in place. An analysis of a year of data (AIRACs 1313 to 1413 (i.e., from the 12th December 2013 until the 07th January 2015)) of ATFM delays has been processed, as described in Section 4.3.

- Background ATFM delay is generated by permuting the actual delay that was generated on the day under study, thus ensuring that the **delay distribution is realistic**. Parameters will be used to model the randomness allowed for this permutation (i.e. the variability with respect to the original delay in the regulation, and also which flights might get assigned a given delay). Hence a small degree of freedom should (practically) re-generate the original scenario of ATFM delay, and higher degrees should increase the number of flights that can potentially have delay assigned but, since this is still a delay permutation of the original, the total distribution of the delay per disturbance type will be maintained.
- The analysis of all the ATFM regulations that were put in place due to the different disturbances modelled in ComplexityCosts ensures that the generation of delay will be **close to that actually obtained** when such disturbances are present in the system. These models will be assessed by testing the delay generated against historic ATFM regulations from a different period of time (i.e. a **hold-out sample**) than that used for the fitting of the generation of probabilities.
- **Historic delay and cancellation series** will be used to assess the credibility of the different disturbance models.

7 Next steps and look ahead

The next steps regarding the model implementation are to:

- Review new classes of actors and new actors instances;
- Revise the new event structure for ComplexityCosts;
- Update cost models and disruptions;
- Run a series of tests.

The next tasks, specifically regarding the mechanisms and disturbances are to:

- complete the specific cost for the different mechanism, to obtain feedback from the consultation with industry, and to model the specific strategic and tactical costs;
- define the ACC(s), airlines and airports that will benefit from the increase ATCo hours, DCI and A-CDM with reaccommodation mechanism and their uptake;
- specify the implementation for the different mechanisms;
- finalise the analysis of ATFM delay to model delay generated by the disturbances;
- create different scenarios with disturbances.

Regarding the delay cost models, the final steps are to:

- await feedback on the airline consultation document and adapt the costs if necessary (see Section 7.4);
- obtain some missing airline, (newly modelled) aircraft and network performance data;
- obtain full 2013 ICAO DATA+ costs (currently partial cost coverage available for 2013; full 2014 costs probably unavailable until 2016);
- update 2014 ICAO DATA+ utilisation data;
- update the statistical reactionary models and publish the reference values.

Next deliverables due are:

Deliverable number	Deliverable ID	Deliverable title	Due date	Comments
D0.08	Progress Report 8	6-monthly Progress Report	15SEP15	
D4.4	SID 2015 contribution	Conference paper to be presented at the Fifth SESAR Innovation Days	30SEP15	Focused on the disturbance modelling ideas presented in Section 4.3 - this is particularly chosen to obtain peer review on this methodology and thus contribute to the wider verification and credibility assessment process described in Section 6.
D0.09	Progress Report 9	Intermediate Progress Report	15DEC15	
D3.1	Scenario definition	Scenario Definition	15DEC15	Including final data requirements and quality control

8 References

Cook, A., Tanner, G., Cristóbal, S., Zanin, M., 2013. New perspectives for air transport performance, D. Schaefer (Ed.), Proceedings of the third SESAR Innovation Days, Stockholm.

Cook, A., Tanner, G., 2011. European airline delay cost reference values, for EUROCONTROL Performance Review Unit, March 2011.

ESG Aviation Services, 2011-2015. Block hour operating costs by aircraft type for the year {2010-2014}, The Airline Monitor.

Deutsche Flugsicherung, 2010. Airport CDM Munich. Results 2009.

Deutsche Flugsicherung, 2013. Airport CDM Munich. Results 2012.

EUROCONTROL, 2008. Airport CDM cost benefit analysis. Ed. 1.4.

EUROCONTROL, 2010. E-OCVM Version 3.0, European Operational Concept Validation Methodology, Volume I.

EUROCONTROL, 2015a. ATM Cost-Effectiveness (ACE) 2013 Benchmarking Report with 2014-2018 outlook.

EUROCONTROL, 2015b. European airport CDM. http://www.euro-cdm.org/ (accessed May 2015).

Gilbert, N, 2008. Agent-based models, Quantitative applications in the social sciences (153), SAGE publications, California, US.

ICAO, 2014. ICAO DATA+, http://www.icao.int/dataplus/ (accessed April 2014), Montréal.

ICAO, 2015. ICAO DATA+, http://www4.icao.int/newdataplus (accessed May-August 2015), Montréal.

NASA, 2010. Technology Readiness Level Definitions - NASA. www.nasa.gov/pdf/458490main_TRL_Definitions.pdf (accessed August 2015).

NATS, 2015. Trainee Air Traffic Controllers - Benefits. http://www.nats.aero/careers/trainee-air-traffic-controllers/benefits/ (accessed August 2015).

SENASA, 2015. En-route and approach air traffic controllers intial training course. http://www.senasa.es/portada.aspx?lang=en-GB&IDPagina=9&IDCurso=92 (accessed August 2015).

SESAR, 2013. E.02.14 - CASSIOPEIA - Study Report: Case Study 3, Deliverable D4.3.

SESAR, 2014. E.02.14 - DCI-4HD2D Dataset Management and case study design, Deliverable D1.1.

SESAR, 2015. SESAR 2020 Exploratory Research: First Call for Research Projects (V 1.1), March 2015.

TRB, 2012. ACRP Report 64: Handbook for Evaluating Emissions and Costs of APUs and Alternative Systems, Washington DC, US.

Appendix A Platform configuration and execution

A.1 Requirements

Although the ComplexityCosts tool is designed to run on a cloud-based infrastructure it is also possible to execute it within a single machine. The minimum requirements to do so are the following:

- Windows XP (x64) SP3, Mac OS 10.9.5, Ubuntu 14.04 LTS, Red Hat Enterprise Linux 6, SUSE Linux Enterprise Desktop 11.3+, Debian 7.x or any higher version of those;
- MATLAB R2013a properly installed and configured;
- Any Intel or AMD x86 processor supporting SSE2 instruction set* or Intel-based Macs with an Intel Core 2 or later;
- At least 3 to 4 GB of disk space;
- 2 GB of RAM.

However, it is worth noting that running the model on these requirements will be possible although not optimal. For example, for the prototype development a set of four c4.2xlarge Amazon EC2 instances were used. Each of these has an Intel Xeon E5-2666 v3 Haswell CPU (10 cores @ 3.3Ghz), 15GB of RAM and a HDD with an optimized throughput of 1000 Mbps.

A.2 Tool deployment

The tool is packaged into a single .zip file containing the latest configuration. It is enough to decompress the files and abiding by the folder structure, main scripts will be added to the root directory. Required databases and additional files are also provided in the single .zip file.

When initializing MATLAB one must change the work directory to where the decompressed files are located (e.g. using the "*cd*" in-line command) and then *genpath(pwd)* follow by *addpath*:

```
> cd('c:\example');
> addpath(genpath(pwd));
```

These commands will generate the folder structure and add it to the current MATLAB path. Once these steps are performed the platform is ready to be executed.

A.3 Setup and configuration

Scenario parameters are defined in the scenario constructor. All of the parameters form a data structure inside the object simulation named parameters. It is structured by a string of characters and cell types. The most important parameters to consider are the following.

Simulation configuration:

```
parameters.simulation.id='configuring CC simulator';
parameters.simulation.numberOfSimulations=1;

parameters.logging.enable=false;

parameters.files.source='.mat'; %'.mat' '.csv
```

The simulation.id names the simulation, which allows the simulator to continue a known simulation and to organize when multiple simulations are executed simultaneously. The numberOfSimulations determines the maximum number of repeated simulations to be performed, since it is not possible to

determine beforehand the number of simulations needed to achieve the desired confidence for the metrics. This number needs to be established to avoid an infinite loop execution of the model.

The logging logical variable determines whether a detailed log will be recorded during each simulation. Recording a log is an extensive processing task and should only be activated when logs are absolutely needed (e.g. debugging).

Loading can be sped up by selecting precomputed .mat files instead of loading .csv files, which can be configured in the files.source parameter.

Sources for the scenario definition are defined in the parameters.files structure, and multiple files are necessary for the execution of the simulation. However, data quality checks are not part of the simulation tool and need to be run again if any of the sources files change.

```
parameters.files.airports='Airports_v1.csv';
parameters.files.traffic='17SEP10_v308b'; % '17SEP10_v308a'   '17SEP10_t100'
parameters.files.pax='17SEP10_PAX_v401'; % '17SEP10_PAX_v401' '17SEP10_PAX_t100'
parameters.files.mtt='SEP10_MTT_v2.mat'; % pre-computed values

parameters.files.airlines='AOdata_v102.csv';
parameters.files.hardpaxcost='AOhardcost_v1.csv';
parameters.files.softpaxcost='AOsoftcost_v1.csv';
parameters.files.nonpaxgate='nonpaxgate_v1.csv';
parameters.files.nonpaxtaxi='nonpaxtaxi_v1.csv';
parameters.files.nonpaxroute='nonpaxroute_v1.csv';
parameters.files.nonpaxarrival='nonpaxarrival_v1.csv';
parameters.files.mct='AirportsMCT_v100.csv';
```

Finally, there is a set of parameters for particular instances and some events behaviour. When no other information is available the defined default values are used; however they can be overwritten at any time during execution.

```
parameters.scenario.inboundThreshold=15/parameters.simulation.timescale;

parameters.pax.hierarchyThreshold=5*60/parameters.simulation.timescale;
parameters.pax.waitingThreshold=60/parameters.simulation.timescale;
parameters.pax.waitingStep=15/parameters.simulation.timescale;
parameters.pax.sentback=0.2;                                        % % of pax that
     do want to go back instead of spend one overnight
parameters.pax.votFlex=50/60;                                       % in euro/min
parameters.pax.votInflex=30/60;                                     % in euro/min

parameters.cost.random=false;

parameters.flights.longhaul=1500;                                   % in NM

parameters.capacity.width=60/parameters.simulation.timescale;       % 60 minutes
     default
parameters.capacity.overlap=15/parameters.simulation.timescale;     % 15 minutes
     default
parameters.capacity.percentile=98.2;                                % capacity
     threshold as taking the maximum would be too risky
parameters.capacity.minimun=45;                                     % mov/h
parameters.capacity.maxDepartureQueue=5/parameters.simulation.timescale;

parameters.turnarround.percentile=2;

parameters.mct.buffer=0;
parameters.mct.std=0.05;
parameters.mct.default=20;

parameters.route.std=0.02;
```

```
parameters.taxi.model='lognorm';                                    % 'invbeta'
     'invgamma', 'norm' 'lognorm'
```

A.4 Initialisation

The model needs to be initialized before execution. First, the meta-classes need to be instantiated using their constructors, and the order does not matter.

```
> driver = simulatorConstructor();
> parameters = scenarioConstructor();
> log = logConstructor();
> results = resultsConstructor();
> eventStack = eventstackConstructor();
```

Secondly, the same process follows for the model entities.

```
> airlines = airlinesConstructor();
> airports = airportsConstructor();
> flights = flightsConstructor();
> pax = paxConstructor();
```

Alternatively, one could use the initModel.m script provided:

```
 > close all
 > clear all

 > addpath(genpath(pwd));

 > initModel;
 Starting a new simulation ...
 Elapsed time is 0.009063 seconds.
 Loading scenario definitions ...
 Elapsed time is 0.016147 seconds.
 Initializing log  ...
 Elapsed time is 0.004062 seconds.
 Loading airline and cost models ...
 Elapsed time is 0.699535 seconds.
 Loading airports database ...
 Elapsed time is 0.126598 seconds.
 Loading traffic ...
 Elapsed time is 7.842945 seconds.
 Updating airport capacities ...
 Elapsed time is 4.375245 seconds.
 Creating connection matrix...
 Elapsed time is 5.984937 seconds.
 Loading pax itineraries information ...
 Elapsed time is 1.906649 seconds.
 Merging pax and flight together ...
 Elapsed time is 32.095750 seconds.
 Configuring simulator ...
 Elapsed time is 0.091728 seconds.
 Lauching Event Manager  ...
 Elapsed time is 0.008170 seconds.
 Ready to simulate!
```

The initModel.m also shows additional information regarding the steps taken and the required time to complete each task.

A.5 Model execution

Once the software is initialized by the initModel script, the meta-object driver is then created. The driver contains the methods to start, continue, restart or stop a simulation. Usually simulations are started from the console using the following command:

```
> driver.simulate();
```

This will start the simulation and produce a control output showing the current simulation time:

```
 > driver.simulate();

 Running simulation 1 of 1 scenario configuring CC simulator ... 17-Sep-2010
08:23:16
```

Once the simulation is completed the elapsed time is displayed and stored for performance analysis. Raw results are stored in the raw format for further analysis.

```
 > driver.simulate();

 Running simulation 1 of 1 scenario configuring CC simulator
 Elapsed time is 98.706904 seconds.
 Writing simulation outputs in file baseline_1.mat ...
 Elapsed time is 5.131558 seconds.
```

A.6 Results and output

Results are stored in the raw mode using the results entity method by default and after each run completion.

```
> results.writeMatlabSingleRun();
```

However, metrics can also be computed without storing the raw results using the following method:

```
> results.computeSingleRunMetrics();
```

In any case results can be save to a .csv file using the following method:

```
> results.saveResults();
```

Results are stored under the folder \results. Each time a new simulation is executed a new folder named as parameters.simulation.ID is created under \results (and \logs) containing all of the saved data (e.g. raw or processed).

-END OF DOCUMENT-